Linguagem MQL5 REFERENTE

ao terminal do cliente MetaTrader 5

ESTUDAR MQL5 e
RESOLVER todas as tarefas:

- Criar seus próprios indicadores de análise técnica com qualquer grau de complexidade
- Utilizar a automatização - automatizar sistemas de negociação para trabalhar em vários mercados financeiros
- Desenvolver suas próprias ferramentas analíticas baseadas em cálculos matemáticos e métodos tradicionais
methods
- Desenvolver sistemas de negociação para resolver uma grande variedade de necessidades(trading, monitoramento, alerta, etc)

© 2000-2024, MetaQuotes Ltd

Content

Referência MQL5 71

1 Elementos Básicos da Linguagem ... 73

Sintaxe ... 74
Comentários ... 75
Identificadores ... 76
Palavras Reservadas ... 77
Tipos de Dados ... 79
Tipos Inteiros ... 80
Tipos char, short, int e long ... 81
Constantes de Caracteres ... 85
Tipo datetime ... 89
Tipo color ... 90
Tipo bool ... 91
Enumeradores ... 92
Tipos Reais (double, float) ... 94
Números complexos (complex) ... 100
Tipo string ... 102
Estruturas, Classes e Interfaces ... 105
Objeto Array Dinâmico ... 132
Matrizes e vetores ... 133
Conversão de Tipo (Typecasting) ... 140
Tipo void e constante NULL ... 145
Tipos personalizados ... 146
Ponteiros de Objeto ... 156
Referência, Modificador & e Palavra-chave this ... 160
Operações e Expressões ... 162
Expressões ... 163
Operações Aritméticas ... 164
Operações de Atribuição ... 165
Operações de Relação ... 166
Operações Booleanas ... 167
Operações Binárias ... 169
Outras Operações ... 172
Regras de Precedência ... 176
Operadores ... 178
Operador Composto ... 179
Operador de Expressão ... 180
Operador return ... 181
Operador Condicional if-else ... 182
Operador Ternário ? ... 183
Operador switch ... 185
Operador de loop while ... 187
Operador de loop for ... 188
Operador de loop do while ... 190
Operador break ... 191
Operador continue ... 192
Operador de Criação de Objeto new ... 193
Operação de Exclusão de Objeto delete ... 195
Funções ... 196
Chamada de Função ... 198
Passando Parâmetros ... 199
Sobrecarga de Função ... 202
Sobrecarga de Operação ... 205

© 2000-2024, MetaQuotes Ltd.

Descrição de Funções Externas ... 219
Exportação de Funções ... 221
Funções de Manipulação de Evento ... 222
Variáveis ... 234
Variáveis Locais ... 238
Parâmetros Formais ... 240
Variáveis Estáticas ... 242
Variáveis Globais ... 244
Variáveis de Entrada ... 245
Variáveis Externas ... 252
Inicialização de Variáveis ... 253
Visibilidade Escopo e Tempo de Vida de Variáveis ... 255
Criação e Exclusão de Objetos ... 257
Preprocessador ... 260
Substituição de Macro (#define) ... 262
Propriedades de Programa (#property) ... 265
Incluindo Arquivos (#include) ... 271
Importando Funções (#import) ... 272
Conditional Compilation (#ifdef, #ifndef, #else, #endif) ... 275
Programação Orientada a Objetos ... 277
Encapsulamento e Extensibilidade de Tipo ... 279
Herança ... 282
Polimorfismo ... 287
Sobrecarga (Overload) ... 291
Funções Virtuais ... 292
Membros Estáticos de uma Classe ... 296
Templates de Função ... 300
Modelos de classes ... 304
Classes abstratas ... 309
Espaços de nomes ... 311
2 Constantes, Enumeradores e Estruturas ... 314
Constantes de Gráfico ... 315
Tipos de Eventos de Gráficos ... 316
Janela temporal de Gráfico ... 323
Propriedades de Gráfico ... 325
Constantes de Posicionamento ... 333
Representação de Gráfico ... 334
Exemplos de como trabalhar com um gráfico ... 336
Constantes de Objetos ... 395
Tipos de Objeto ... 396
OBJ_VLINE ... 398
OBJ_HLINE ... 403
OBJ_TREND ... 408
OBJ_TRENDBYANGLE ... 415
OBJ_CYCLES ... 421
OBJ_ARROWED_LINE ... 427
OBJ_CHANNEL ... 433
OBJ_STDDEVCHANNEL ... 440
OBJ_REGRESSION ... 447
OBJ_PITCHFORK ... 453
OBJ_GANNLINE ... 461
OBJ_GANNFAN ... 468
OBJ_GANNGRID ... 475
OBJ_FIBO ... 482
OBJ_FIBOTIMES ... 489
OBJ_FIBOFAN ... 496
OBJ_FIBOARC ... 503
OBJ_FIBOCHANNEL ... 510

© 2000-2024, MetaQuotes Ltd.

OBJ_EXPANSION ... 518
OBJ_ELLIOTWAVE5 ... 526
OBJ_ELLIOTWAVE3 ... 534
OBJ_RECTANGLE ... 541
OBJ_TRIANGLE ... 547
OBJ_ELLIPSE ... 554
OBJ_ARROW_THUMB_UP ... 561
OBJ_ARROW_THUMB_DOWN ... 567
OBJ_ARROW_UP ... 573
OBJ_ARROW_DOWN ... 579
OBJ_ARROW_STOP ... 585
OBJ_ARROW_CHECK ... 591
OBJ_ARROW_LEFT_PRICE ... 597
OBJ_ARROW_RIGHT_PRICE ... 602
OBJ_ARROW_BUY ... 607
OBJ_ARROW_SELL ... 612
OBJ_ARROW ... 617
OBJ_TEXT ... 623
OBJ_LABEL ... 629
OBJ_BUTTON ... 637
OBJ_CHART ... 644
OBJ_BITMAP ... 651
OBJ_BITMAP_LABEL ... 658
OBJ_EDIT ... 665
OBJ_EVENT ... 672
OBJ_RECTANGLE_LABEL ... 677
Propriedades do Objeto ... 683
Métodos de Vinculação de Objeto ... 711
Canto de Gráfico ... 716
Visibilidade de Objetos ... 719
Níveis de Onda de Elliott ... 722
Objetos Gann ... 723
Cores Web ... 725
Wingdings ... 727
Constantes de Indicador ... 728
Constantes de Preço ... 729
Métodos de Suavização ... 732
Linhas de Indicadores ... 733
Estilos de Desenho ... 735
Propriedades de Indicador Customizado ... 741
Tipos de Indicador ... 747
Identificadores de Tipo de Dados ... 749
Estado de Ambiente ... 750
Propriedades do Terminal Cliente ... 751
Propriedades de um Programa MQL5 em Execução ... 757
Propriedades do Ativo ... 761
Propriedades da Conta ... 882
Estatísticas de Teste ... 893
Constantes de Negociação ... 898
Propriedades de uma Base Histórica de Dados ... 899
Propriedades de uma Ordem ... 900
Propriedades de uma Posição ... 919
Propriedades de uma Operação ... 923
Tipos de Operações de Negociação ... 927
Tipos de Transação de Negociação ... 939
Ordens de Negociação em DOM (Profundidade de Mercado) ... 942
Propriedades de Sinal ... 943
Constantes Nomeados ... 945

© 2000-2024, MetaQuotes Ltd.

Substituições de Macro Predefinidas ... 946
Constantes Matemáticas ... 952
Constantes de Tipo Numérico ... 954
Códigos de Motivos de Desinicialização ... 957
Verificação de Ponteiro de Objeto ... 959
Outras Constantes ... 960
Estruturas de Dados ... 964
Estrutura do Tipo Data ... 965
Estrutura de Parâmetros de Entrada ... 966
Estrutura de Dados Históricos ... 967
Estrutura DOM ... 968
Estrutura de Solicitação de Negociação (Trade) ... 969
Estrutura de Resultados de Verificação de Solicitação ... 982
Estrutura de Resultado de Solicitação de uma Negociação (Trade) ... 983
Estrutura de uma Transação de Negociação ... 987
Estrutura para Preços Correntes ... 995
Estruturas do calendário econômico ... 997
Códigos de Erros e Avisos ... 1006
Códigos de Retorno do Servidor de Negociação ... 1007
Avisos do Compilador ... 1011
Erros de Compilação ... 1014
Erros em Tempo de Execução ... 1025
Constantes de Entrada/Saída ... 1039
Flags de Abertura de Arquivo ... 1040
Propriedades de Arquivo ... 1042
Posição In-File ... 1043
Uso de um Código de Página (Codepage) ... 1044
MessageBox ... 1045
3 Programas MQL5 ... 1047
Execução de Programa ... 1048
Trade permission ... 1055
Eventos do Terminal Cliente ... 1059
Recursos ... 1062
Chamadas de Funções Importadas ... 1074
Erros em Tempo de Execução ... 1076
Testando Estratégias de Negociação ... 1077
4 Variáveis Predefinidas ... 1105
_AppliedTo ... 1106
_Digits ... 1108
_Point ... 1109
_LastError ... 1110
_Period ... 1111
_RandomSeed ... 1112
_StopFlag ... 1113
_Symbol ... 1114
_UninitReason ... 1115
_IsX64 ... 1116
5 Funções Comuns ... 1117
Alert ... 1119
CheckPointer ... 1120
Comment ... 1122
CryptEncode ... 1124
CryptDecode ... 1126
DebugBreak ... 1127
ExpertRemove ... 1128
GetPointer ... 1130
GetTickCount ... 1134

© 2000-2024, MetaQuotes Ltd.

GetTickCount64 ... 1135
GetMicrosecondCount ... 1136
MessageBox ... 1138
PeriodSeconds ... 1139
PlaySound ... 1140
Print ... 1141
PrintFormat ... 1143
ResetLastError ... 1149
ResourceCreate ... 1150
ResourceFree ... 1152
ResourceReadImage ... 1153
ResourceSave ... 1154
SetReturnError ... 1155
SetUserError ... 1156
Sleep ... 1157
TerminalClose ... 1158
TesterHideIndicators ... 1160
TesterStatistics ... 1162
TesterStop ... 1163
TesterDeposit ... 1164
TesterWithdrawal ... 1165
TranslateKey ... 1166
ZeroMemory ... 1167

6 Funções para Array ... 1168

ArrayBsearch ... 1169
ArrayCopy ... 1173
ArrayCompare ... 1178
ArrayFree ... 1179
ArrayGetAsSeries ... 1188
ArrayInitialize ... 1191
ArrayFill ... 1193
ArrayIsDynamic ... 1195
ArrayIsSeries ... 1197
ArrayMaximum ... 1199
ArrayMinimum ... 1210
ArrayPrint ... 1221
ArrayRange ... 1224
ArrayResize ... 1225
ArrayInsert ... 1228
ArrayRemove ... 1231
ArrayReverse ... 1233
ArraySetAsSeries ... 1235
ArraySize ... 1238
ArraySort ... 1240
ArraySwap ... 1245

7 Métodos matriciais e vetoriais ... 1247

Tipos de matrizes e vetores ... 1254
Enumerações ... 1256
Inicialização ... 1261
Assign ... 1264
CopyIndicatorBuffer ... 1266
CopyRates ... 1268
CopyTicks ... 1273
CopyTicksRange ... 1276
Eye ... 1278
Identity ... 1280
Ones ... 1282
Zeros ... 1283

© 2000-2024, MetaQuotes Ltd.

Full ... 1284
Tri ... 1285
Init ... 1286
Fill ... 1288
Atribuições ... 1289
HasNan ... 1290
Transpose ... 1291
TriL ... 1293
TriU ... 1294
Diag ... 1295
Row ... 1297
Col ... 1299
Copy ... 1301
Compare ... 1303
CompareByDigits ... 1305
Flat ... 1307
Clip ... 1309
Reshape ... 1310
Resize ... 1312
Set ... 1314
SwapRows ... 1316
SwapCols ... 1317
Split ... 1318
Hsplit ... 1320
Vsplit ... 1322
ArgSort ... 1324
Sort ... 1325
Operações ... 1327
Operações matemáticas ... 1329
Funções matemáticas ... 1330
Produtos ... 1331
MatMul ... 1332
GeMM ... 1337
Power ... 1341
Dot ... 1344
Kron ... 1346
Inner ... 1348
Outer ... 1350
CorrCoef ... 1353
Cov ... 1357
Correlate ... 1360
Convolve ... 1363
Conversões ... 1366
Cholesky ... 1367
Eig ... 1368
EigVals ... 1371
LU ... 1372
LUP ... 1374
QR ... 1376
SVD ... 1378
Estatísticas ... 1381
ArgMax ... 1382
ArgMin ... 1383
Max ... 1384
Min ... 1385
Ptp ... 1386
Sum ... 1387
Prod ... 1388

© 2000-2024, MetaQuotes Ltd.

CumSum ... 1390
CumProd ... 1392
Percentile ... 1394
Quantile ... 1396
Median ... 1398
Mean ... 1400
Average ... 1401
Std ... 1403
Var ... 1405
LinearRegression ... 1407
Características ... 1410
Rows ... 1411
Cols ... 1412
Size ... 1413
Norm ... 1414
Cond ... 1417
Det ... 1420
SLogDet ... 1422
Rank ... 1423
Trace ... 1426
Spectrum ... 1427
Soluções ... 1428
Solve ... 1429
LstSq ... 1430
Inv ... 1431
PInv ... 1433
Aprendizado de máquina ... 1434
Activation ... 1440
Derivative ... 1444
Loss ... 1446
LossGradient ... 1448
RegressionMetric ... 1450
ConfusionMatrix ... 1452
ConfusionMatrixMultilabel ... 1454
ClassificationMetric ... 1456
ClassificationScore ... 1459
PrecisionRecall ... 1463
ReceiverOperatingCharacteristic ... 1467
8 Funções de Conversão ... 1471
CharToString ... 1473
CharArrayToString ... 1474
CharArrayToStruct ... 1475
StructToCharArray ... 1476
ColorToARGB ... 1477
ColorToString ... 1479
DoubleToString ... 1480
EnumToString ... 1481
IntegerToString ... 1483
ShortToString ... 1484
ShortArrayToString ... 1485
TimeToString ... 1486
NormalizeDouble ... 1487
StringToCharArray ... 1489
StringToColor ... 1490
StringToDouble ... 1491
StringToInteger ... 1492
StringToShortArray ... 1493
StringToTime ... 1494

© 2000-2024, MetaQuotes Ltd.

StringFormat ... 1495
9 Funções Matemáticas ... 1499
MathAbs ... 1501
MathArccos ... 1502
MathArcsin ... 1503
MathArctan ... 1504
MathArctan2 ... 1505
MathClassify ... 1506
MathCeil ... 1508
MathCos ... 1509
MathExp ... 1510
MathFloor ... 1511
MathLog ... 1512
MathLog10 ... 1513
MathMax ... 1514
MathMin ... 1515
MathMod ... 1516
MathPow ... 1517
MathRand ... 1518
MathRound ... 1519
MathSin ... 1520
MathSqrt ... 1521
MathSrand ... 1522
MathTan ... 1525
MathIsValidNumber ... 1526
MathExpm1 ... 1527
MathLog1p ... 1528
MathArccosh ... 1529
MathArcsinh ... 1530
MathArctanh ... 1531
MathCosh ... 1532
MathSinh ... 1533
MathTanh ... 1534
MathSwap ... 1535
10 Funções de String ... 1536
StringAdd ... 1537
StringBufferLen ... 1539
StringCompare ... 1540
StringConcatenate ... 1542
StringFill ... 1543
StringFind ... 1544
StringGetCharacter ... 1545
StringInit ... 1546
StringLen ... 1547
StringSetLength ... 1548
StringReplace ... 1549
StringReserve ... 1550
StringSetCharacter ... 1552
StringSplit ... 1554
StringSubstr ... 1556
StringToLower ... 1557
StringToUpper ... 1558
StringTrimLeft ... 1559
StringTrimRight ... 1560
11 Data e Hora ... 1561
TimeCurrent ... 1562
TimeTradeServer ... 1563

© 2000-2024, MetaQuotes Ltd.

TimeLocal 1564
TimeGMT 1565
TimeDaylightSavings 1566
TimeGMTOffset 1567
TimeToStruct 1568
StructToTime 1569
12 Informações de Conta 1570
AccountInfoDouble 1571
AccountInfoInteger 1572
AccountInfoString 1574
13 Verificando Estado 1575
GetLastError 1576
IsStopped 1577
UninitializeReason 1578
TerminalInfoInteger 1579
TerminalInfoDouble 1580
TerminalInfoString 1581
MQLInfoInteger 1582
MQLInfoString 1583
Symbol 1584
Period 1585
Digits 1586
Point 1587
14 Manipulação de eventos 1588
OnStart 1590
OnInit 1593
OnDeinit 1596
OnTick 1599
OnCalculate 1605
OnTimer 1609
OnTrade 1612
OnTradeTransaction 1617
OnBookEvent 1623
OnChartEvent 1627
OnTester 1634
OnTesterInit 1641
OnTesterDeinit 1648
OnTesterPass 1649
15 Informações de Mercado 1650
SymbolsTotal 1651
SymbolExist 1652
SymbolName 1653
SymbolSelect 1654
SymbolIsSynchronized 1655
SymbolInfoDouble 1656
SymbolInfoInteger 1658
SymbolInfoString 1660
SymbolInfoMarginRate 1661
SymbolInfoTick 1662
SymbolInfoSessionQuote 1663
SymbolInfoSessionTrade 1664
MarketBookAdd 1665
MarketBookRelease 1666
MarketBookGet 1667
16 Calendário econômico 1668
CalendarCountryById 1669

© 2000-2024, MetaQuotes Ltd.

CalendarEventById 1671
CalendarValueById 1674
CalendarCountries 1677
CalendarEventByCountry 1679
CalendarEventByCurrency 1681
CalendarValueHistoryByEvent 1683
CalendarValueHistory 1687
CalendarValueLastByEvent 1690
CalendarValueLast 1695

17 Séries Temporais e Acesso a Indicadores 1700

Direção de Indexação em Arrays, Buffers e Séries Temporais 1705
Organizando Acesso aos Dados 1709
SeriesInfoInteger 1719
Bars 1721
BarsCalculated 1724
IndicatorCreate 1726
IndicatorParameters 1728
IndicatorRelease 1730
CopyBuffer 1732
CopyRates 1737
CopySeries 1741
CopyTime 1745
CopyOpen 1748
CopyHigh 1751
CopyLow 1755
CopyClose 1758
CopyTickVolume 1761
CopyRealVolume 1765
CopySpread 1768
CopyTicks 1772
CopyTicksRange 1777
iBars 1779
iBarShift 1780
iClose 1783
iHigh 1785
iHighest 1787
iLow 1788
iLowest 1790
iOpen 1791
iTime 1793
iTickVolume 1795
iRealVolume 1797
iVolume 1799
iSpread 1801

18 Símbolos personalizados 1803

CustomSymbolCreate 1805
CustomSymbolDelete 1807
CustomSymbolSetInteger 1808
CustomSymbolSetDouble 1809
CustomSymbolSetString 1810
CustomSymbolSetMarginRate 1811
CustomSymbolSetSessionQuote 1812
CustomSymbolSetSessionTrade 1813
CustomRatesDelete 1814
CustomRatesReplace 1815
CustomRatesUpdate 1816
CustomTicksAdd 1817
CustomTicksDelete 1819

© 2000-2024, MetaQuotes Ltd.

CustomTicksReplace 1820
CustomBookAdd 1822
19 Operações de Gráficos.... 1825
ChartApplyTemplate 1827
ChartSaveTemplate 1830
ChartWindowFind 1835
ChartTimePriceToXY 1837
ChartXYToTimePrice 1838
ChartOpen 1840
ChartFirst 1841
ChartNext 1842
ChartClose 1843
ChartSymbol 1844
ChartPeriod 1845
ChartRedraw 1846
ChartSetDouble 1847
ChartSetInteger 1848
ChartSetString 1850
ChartGetDouble 1852
ChartGetInteger 1854
ChartGetString 1856
ChartNavigate 1858
ChartID 1861
ChartIndicatorAdd 1862
ChartIndicatorDelete 1866
ChartIndicatorGet 1869
ChartIndicatorName 1871
ChartIndicatorsTotal 1872
ChartWindowOnDropped 1873
ChartPriceOnDropped 1874
ChartTimeOnDropped 1875
ChartXOnDropped 1876
ChartYOnDropped 1877
ChartSetSymbolPeriod 1878
ChartScreenShot 1879
20 Funções de Negociação.... 1882
OrderCalcMargin 1884
OrderCalcProfit 1885
OrderCheck 1886
OrderSend 1887
OrderSendAsync 1892
PositionsTotal 1903
PositionGetSymbol 1904
PositionSelect 1905
PositionSelectByTicket 1906
PositionGetDouble 1907
PositionGetInteger 1908
PositionGetTicket 1910
PositionGetString 1911
OrdersTotal 1912
OrderGetTicket 1913
OrderSelect 1915
OrderGetDouble 1916
OrderGetInteger 1917
OrderGetString 1918
HistorySelect 1919
HistorySelectByPosition 1921
HistoryOrderSelect 1922

© 2000-2024, MetaQuotes Ltd.

HistoryOrdersTotal ... 1923
HistoryOrderGetTicket ... 1924
HistoryOrderGetDouble ... 1926
HistoryOrderGetInteger ... 1927
HistoryOrderGetString ... 1930
HistoryDealSelect ... 1931
HistoryDealsTotal ... 1932
HistoryDealGetTicket ... 1933
HistoryDealGetDouble ... 1935
HistoryDealGetInteger ... 1936
HistoryDealGetString ... 1939

21 Gerenciamento de sinais ... 1940

SignalBaseGetDouble ... 1941
SignalBaseGetInteger ... 1942
SignalBaseGetString ... 1943
SignalBaseSelect ... 1944
SignalBaseTotal ... 1945
SignalInfoGetDouble ... 1946
SignalInfoGetInteger ... 1947
SignalInfoGetString ... 1948
SignalInfoSetDouble ... 1949
SignalInfoSetInteger ... 1950
SignalSubscribe ... 1951
SignalUnsubscribe ... 1952

22 Funções de rede ... 1953

SocketCreate ... 1955
SocketClose ... 1958
SocketConnect ... 1961
SocketIsConnected ... 1965
SocketIsReadable ... 1966
SocketIsWritable ... 1969
SocketTimeouts ... 1970
SocketRead ... 1971
SocketSend ... 1975
SocketTlsHandshake ... 1979
SocketTlsCertificate ... 1980
SocketTlsRead ... 1984
SocketTlsReadAvailable ... 1988
SocketTlsSend ... 1989
WebRequest ... 1990
SendFTP ... 1993
SendMail ... 1994
SendNotification ... 1995

23 Variáveis Globais do Terminal ... 1996

GlobalVariableCheck ... 1997
GlobalVariableTime ... 1998
GlobalVariableDel ... 1999
GlobalVariableGet ... 2000
GlobalVariableName ... 2001
GlobalVariableSet ... 2002
GlobalVariablesFlush ... 2003
GlobalVariableTemp ... 2004
GlobalVariableSetOnCondition ... 2005
GlobalVariablesDeleteAll ... 2006
GlobalVariablesTotal ... 2007

24 Funções de Arquivo ... 2008

FileSelectDialog ... 2011

© 2000-2024, MetaQuotes Ltd.

FileFindFirst 2013
FileFindNext 2015
FileFindClose 2017
FileIsExist 2019
FileOpen 2022
FileClose 2025
FileCopy 2026
FileDelete 2029
FileMove 2031
FileFlush 2034
FileGetInteger 2036
FileIsEnding 2039
FileIsLineEnding 2041
FileReadArray 2046
FileReadBool 2048
FileReadDatetime 2051
FileReadDouble 2054
FileReadFloat 2057
FileReadInteger 2060
FileReadLong 2064
FileReadNumber 2067
FileReadString 2072
FileReadStruct 2074
FileSeek 2078
FileSize 2081
FileTell 2083
FileWrite 2086
FileWriteArray 2089
FileWriteDouble 2092
FileWriteFloat 2095
FileWriteInteger 2097
FileWriteLong 2100
FileWriteString 2102
FileWriteStruct 2105
FileLoad 2108
FileSave 2110
FolderCreate 2112
FolderDelete 2115
FolderClean 2118

25 Indicadores Customizados 2121

Estilos de Indicador em Exemplos 2125
DRAW_NONE 2136
DRAW_LINE 2139
DRAW_SECTION 2143
DRAW_HISTOGRAM 2147
DRAW_HISTOGRAM2 2151
DRAW_ARROW 2155
DRAW_ZIGZAG 2160
DRAW_FILLING 2165
DRAW_BARS 2170
DRAW_CANDLES 2176
DRAW_COLOR_LINE 2183
DRAW_COLOR_SECTION 2188
DRAW_COLOR_HISTOGRAM 2194
DRAW_COLOR_HISTOGRAM2 2199
DRAW_COLOR_ARROW 2204
DRAW_COLOR_ZIGZAG 2210
DRAW_COLOR_BARS 2215

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_CANDLES ... 2222
Conexão entre Propriedades do Indicador e Funções ... 2229
SetIndexBuffer ... 2232
IndicatorSetDouble ... 2235
IndicatorSetInteger ... 2239
IndicatorSetString ... 2243
PlotIndexSetDouble ... 2246
PlotIndexSetInteger ... 2247
PlotIndexSetString ... 2251
PlotIndexGetInteger ... 2252
26 Funções de Objeto ... 2255
ObjectCreate ... 2257
ObjectName ... 2261
ObjectDelete ... 2262
ObjectsDeleteAll ... 2263
ObjectFind ... 2264
ObjectGetTimeByValue ... 2265
ObjectGetValueByTime ... 2266
ObjectMove ... 2267
ObjectsTotal ... 2268
ObjectSetDouble ... 2269
ObjectSetInteger ... 2273
ObjectSetString ... 2276
ObjectGetDouble ... 2278
ObjectGetInteger ... 2280
ObjectGetString ... 2282
TextSetFont ... 2284
TextOut ... 2287
TextGetSize ... 2291
27 Indicadores Técnicos ... 2292
iAC ... 2295
iAD ... 2300
iADX ... 2305
iADXWilder ... 2310
iAlligator ... 2315
iAMA ... 2322
iAO ... 2327
iATR ... 2332
iBearsPower ... 2337
iBands ... 2342
iBullsPower ... 2348
iCCI ... 2353
iChaikin ... 2358
iCustom ... 2363
iDEMA ... 2366
iDeMarker ... 2371
iEnvelopes ... 2376
iForce ... 2382
iFractals ... 2387
iFrAMA ... 2392
iGator ... 2397
iIchimoku ... 2404
iBWMFI ... 2411
iMomentum ... 2416
iMFI ... 2421
iMA ... 2426
iOsMA ... 2431
iMACD ... 2436

© 2000-2024, MetaQuotes Ltd.

iOBV 2442
iSAR 2447
iRSI 2452
iRVI 2457
iStdDev 2462
iStochastic 2467
iTEMA 2473
iTriX 2478
iWPR 2483
iVIDyA 2488
iVolumes 2493
28 Trabalhando com Resultados de Otimização.... 2498
FrameFirst 2500
FrameFilter 2501
FrameNext 2502
FrameInputs 2503
FrameAdd 2504
ParameterGetRange 2505
ParameterSetRange 2508
29 Trabalhando com Eventos.... 2510
EventSetMillisecondTimer 2511
EventSetTimer 2512
EventKillTimer 2513
EventChartCustom 2514
30 Trabalhando com OpenCL.... 2520
CLHandleType 2522
CLGetInfoInteger 2523
CLGetInfoString 2526
CLContextCreate 2529
CLContextFree 2530
CLGetDeviceInfo 2531
CLProgramCreate 2536
CLProgramFree 2540
CLKernelCreate 2541
CLKernelFree 2542
CLSetKernelArg 2543
CLSetKernelArgMem 2544
CLSetKernelArgMemLocal 2545
CLBufferCreate 2546
CLBufferFree 2547
CLBufferWrite 2548
CLBufferRead 2553
CLExecute 2557
CLExecutionStatus 2559
31 Trabalhar com bancos de dados 2560
DatabaseOpen 2563
DatabaseClose 2565
DatabaseImport 2566
DatabaseExport 2569
DatabasePrint 2575
DatabaseTableExists 2580
DatabaseExecute 2581
DatabasePrepare 2593
DatabaseReset 2602
DatabaseBind 2608
DatabaseBindArray 2613
DatabaseRead 2618

© 2000-2024, MetaQuotes Ltd.

DatabaseReadBind 2619
DatabaseFinalize 2623
DatabaseTransactionBegin 2624
DatabaseTransactionCommit 2629
DatabaseTransactionRollback 2630
DatabaseColumnsCount 2631
DatabaseColumnName 2632
DatabaseColumnType 2633
DatabaseColumnSize 2634
DatabaseColumnText 2635
DatabaseColumnInteger 2636
DatabaseColumnLong 2637
DatabaseColumnDouble 2638
DatabaseColumnBlob 2639

32 Trabalho com DirectX 2640

DXContextCreate 2642
DXContextSetSize 2643
DXContextGetSize 2644
DXContextClearColors 2645
DXContextClearDepth 2646
DXContextGetColors 2647
DXContextGetDepth 2648
DXBufferCreate 2649
DXTextureCreate 2650
DXInputCreate 2656
DXInputSet 2657
DXShaderCreate 2658
DXShaderSetLayout 2659
DXShaderInputsSet 2660
DXShaderTexturesSet 2661
DXDraw 2662
DXDrawIndexed 2663
DXPrimiveTopologySet 2664
DXBufferSet 2665
DXShaderSet 2666
DXHandleType 2667
DXRelease 2668

33 MetaTrader para Python 2669

initialize 2675
login 2677
shutdown 2680
version 2681
last_error 2683
account_info 2685
terminal_info 2688
symbols_total 2691
symbols_get 2692
symbol_info 2695
symbol_info_tick 2699
symbol_select 2701
market_book_add 2705
market_book_get 2706
market_book_release 2709
copy_rates_from 2710
copy_rates_from_pos 2714
copy_rates_range 2717
copy_ticks_from 2720
copy_ticks_range 2724

© 2000-2024, MetaQuotes Ltd.

orders_total 2727
orders_get 2728
order_calc_margin 2731
order_calc_profit 2734
order_check 2737
order_send 2741
positions_total 2746
positions_get 2747
history_orders_total 2750
history_orders_get 2752
history_deals_total 2755
history_deals_get 2757
34 Modelos ONNX 2761
Suporte a ONNX 2762
Conversão de formatos 2764
Autoconversão de dados 2765
Criação de modelo 2769
Inicialização do modelo 2777
Verificação no testador 2783
OnnxCreate 2788
OnnxCreateFromBuffer 2789
OnnxRelease 2790
OnnxRun 2791
OnnxGetInputCount 2794
OnnxGetOutputCount 2795
OnnxGetInputName 2796
OnnxGetOutputName 2797
OnnxGetInputTypeInfo 2798
OnnxGetOutputTypeInfo 2799
OnnxSetInputShape 2800
OnnxSetOutputShape 2802
Estruturas de dados 2804
35 Biblioteca Padrão 2807
Matemática 2808
Estatística 2809
Características estatísticas 2812
MathMean 2813
MathVariance 2814
MathSkewness 2815
MathKurtosis 2816
MathMoments 2817
MathMedian 2818
MathStandardDeviation 2819
MathAverageDeviation 2820
Distribuição normal 2821
MathProbabilityDensityNormal 2825
MathCumulativeDistributionNormal 2827
MathQuantileNormal 2829
MathRandomNormal 2831
MathMomentsNormal 2832
Distribuição log-normal 2833
MathProbabilityDensityLognormal 2837
MathCumulativeDistributionLognormal 2839
MathQuantileLognormal 2841
MathRandomLognormal 2843
MathMomentsLognormal 2844
Distribuição beta 2845
MathProbabilityDensityBeta 2849

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBeta ... 2851
MathQuantileBeta ... 2853
MathRandomBeta ... 2855
MathMomentsBeta ... 2856
Distribuição beta não-central ... 2857
MathProbabilityDensityNoncentralBeta ... 2861
MathCumulativeDistributionNoncentralBeta ... 2863
MathQuantileNoncentralBeta ... 2865
MathRandomNoncentralBeta ... 2867
MathMomentsNoncentralBeta ... 2868
Distribuição gama ... 2869
MathProbabilityDensityGamma ... 2873
MathCumulativeDistributionGamma ... 2875
MathQuantileGamma ... 2877
MathRandomGamma ... 2879
MathMomentsGamma ... 2880
Qui-quadrado ... 2881
MathProbabilityDensityChiSquare ... 2885
MathCumulativeDistributionChiSquare ... 2887
MathQuantileChiSquare ... 2889
MathRandomChiSquare ... 2891
MathMomentsChiSquare ... 2892
Qui-quadrado não-central ... 2893
MathProbabilityDensityNoncentralChiSquare ... 2897
MathCumulativeDistributionNoncentralChiSquare ... 2899
MathQuantileNoncentralChiSquare ... 2901
MathRandomNoncentralChiSquare ... 2903
MathMomentsNoncentralChiSquare ... 2904
Distribuição exponencial ... 2905
MathProbabilityDensityExponential ... 2909
MathCumulativeDistributionExponential ... 2911
MathQuantileExponential ... 2913
MathRandomExponential ... 2915
MathMomentsExponential ... 2916
Distribuição F de Fisher ... 2917
MathProbabilityDensityF ... 2921
MathCumulativeDistributionF ... 2923
MathQuantileF ... 2925
MathRandomF ... 2927
MathMomentsF ... 2928
Distribuição F de Fisher não-central ... 2929
MathProbabilityDensityNoncentralF ... 2933
MathCumulativeDistributionNoncentralF ... 2935
MathQuantileNoncentralF ... 2937
MathRandomNoncentralF ... 2939
MathMomentsNoncentralF ... 2940
Distribuição t de Student ... 2941
MathProbabilityDensityT ... 2945
MathCumulativeDistributionT ... 2947
MathQuantileT ... 2949
MathRandomT ... 2951
MathMomentsT ... 2952
Distribuição t de Student não-central ... 2953
MathProbabilityDensityNoncentralT ... 2957
MathCumulativeDistributionNoncentralT ... 2959
MathQuantileNoncentralT ... 2961
MathRandomNoncentralT ... 2963
MathMomentsNoncentralT ... 2964

© 2000-2024, MetaQuotes Ltd.

Distribuição logística ... 2965
MathProbabilityDensityLogistic ... 2969
MathCumulativeDistributionLogistic ... 2971
MathQuantileLogistic ... 2973
MathRandomLogistic ... 2975
MathMomentsLogistic ... 2976
Distribuição de Cauchy ... 2977
MathProbabilityDensityCauchy ... 2981
MathCumulativeDistributionCauchy ... 2983
MathQuantileCauchy ... 2985
MathRandomCauchy ... 2987
MathMomentsCauchy ... 2988
Distribuição uniforme ... 2989
MathProbabilityDensityUniform ... 2993
MathCumulativeDistributionUniform ... 2995
MathQuantileUniform ... 2997
MathRandomUniform ... 2999
MathMomentsUniform ... 3000
Distribuição de Weibull ... 3001
MathProbabilityDensityWeibull ... 3005
MathCumulativeDistributionWeibull ... 3007
MathQuantileWeibull ... 3009
MathRandomWeibull ... 3011
MathMomentsWeibull ... 3012
Distribuição binomial ... 3013
MathProbabilityDensityBinomial ... 3016
MathCumulativeDistributionBinomial ... 3018
MathQuantileBinomial ... 3020
MathRandomBinomial ... 3022
MathMomentsBinomial ... 3023
Distribuição binomial negativa ... 3024
MathProbabilityDensityNegativeBinomial ... 3027
MathCumulativeDistributionNegativeBinomial ... 3029
MathQuantileNegativeBinomial ... 3031
MathRandomNegativeBinomial ... 3033
MathMomentsNegativeBinomial ... 3034
Distribuição geométrica ... 3035
MathProbabilityDensityGeometric ... 3039
MathCumulativeDistributionGeometric ... 3041
MathQuantileGeometric ... 3043
MathRandomGeometric ... 3045
MathMomentsGeometric ... 3046
Distribuição hipergeométrica ... 3047
MathProbabilityDensityHypergeometric ... 3051
MathCumulativeDistributionHypergeometric ... 3053
MathQuantileHypergeometric ... 3055
MathRandomHypergeometric ... 3057
MathMomentsHypergeometric ... 3058
Distribuição de Poisson ... 3059
MathProbabilityDensityPoisson ... 3063
MathCumulativeDistributionPoisson ... 3065
MathQuantilePoisson ... 3067
MathRandomPoisson ... 3069
MathMomentsPoisson ... 3070
Funções auxiliares ... 3071
MathRandomNonZero ... 3076
MathMoments ... 3077
MathPowInt ... 3078

© 2000-2024, MetaQuotes Ltd.

MathFactorial ... 3079
MathTrunc ... 3080
MathRound ... 3081
MathArctan2 ... 3083
MathGamma ... 3085
MathGammaLog ... 3086
MathBeta ... 3087
MathBetaLog ... 3088
MathBetaIncomplete ... 3089
MathGammaIncomplete ... 3090
MathBinomialCoefficient ... 3091
MathBinomialCoefficientLog ... 3092
MathHypergeometric2F2 ... 3093
MathSequence ... 3094
MathSequenceByCount ... 3095
MathReplicate ... 3096
MathReverse ... 3097
MathIdentical ... 3098
MathUnique ... 3099
MathQuickSortAscending ... 3100
MathQuickSortDescending ... 3101
MathQuickSort ... 3102
MathOrder ... 3103
MathBitwiseNot ... 3104
MathBitwiseAnd ... 3105
MathBitwiseOr ... 3106
MathBitwiseXor ... 3107
MathBitwiseShiftL ... 3108
MathBitwiseShiftR ... 3109
MathCumulativeSum ... 3110
MathCumulativeProduct ... 3111
MathCumulativeMin ... 3112
MathCumulativeMax ... 3113
MathSin ... 3114
MathCos ... 3115
MathTan ... 3116
MathArcsin ... 3117
MathArccos ... 3118
MathArctan ... 3119
MathSinPi ... 3120
MathCosPi ... 3121
MathTanPi ... 3122
MathAbs ... 3123
MathCeil ... 3124
MathFloor ... 3125
MathSqrt ... 3126
MathExp ... 3127
MathPow ... 3128
MathLog ... 3129
MathLog2 ... 3130
MathLog10 ... 3131
MathLog1p ... 3132
MathDifference ... 3133
MathSample ... 3135
MathTukeySummary ... 3138
MathRange ... 3139
MathMin ... 3140
MathMax ... 3141

© 2000-2024, MetaQuotes Ltd.

MathSum ... 3142
MathProduct ... 3143
MathStandardDeviation ... 3144
MathAverageDeviation ... 3145
MathMedian ... 3146
MathMean ... 3147
MathVariance ... 3148
MathSkewness ... 3149
MathKurtosis ... 3150
MathExpm1 ... 3151
MathSinh ... 3152
MathCosh ... 3153
MathTanh ... 3154
MathArcsinh ... 3155
MathArccosh ... 3156
MathArctanh ... 3157
MathSignif ... 3158
MathRank ... 3160
MathCorrelationPearson ... 3161
MathCorrelationSpearman ... 3162
MathCorrelationKendall ... 3163
MathQuantile ... 3164
MathProbabilityDensityEmpirical ... 3165
MathCumulativeDistributionEmpirical ... 3166
Lógica difusa ... 3167
Funções de associação ... 3169
CConstantMembershipFunction ... 3171
GetValue ... 3173
CCompositeMembershipFunction ... 3174
CompositionType ... 3176
MembershipFunctions ... 3176
GetValue ... 3176
CDifferencTwoSigmoidalMembershipFunction ... 3178
A1 ... 3180
A2 ... 3180
C1 ... 3181
C2 ... 3181
GetValue ... 3182
CGeneralizedBellShapedMembershipFunction ... 3183
A ... 3185
B ... 3185
C ... 3186
GetValue ... 3186
CNormalCombinationMembershipFunction ... 3187
B1 ... 3189
B2 ... 3189
Sigma1 ... 3190
Sigma2 ... 3190
GetValue ... 3191
CNormalMembershipFunction ... 3192
B ... 3194
Sigma ... 3194
GetValue ... 3195
CP_ShapedMembershipFunction ... 3196
A ... 3198
B ... 3198
C ... 3199
D ... 3199

© 2000-2024, MetaQuotes Ltd.

GetValue ... 3200
CProductTwoSigmoidalMembershipFunctions ... 3201
A1 ... 3203
A2 ... 3203
C1 ... 3204
C2 ... 3204
GetValue ... 3205
CS_ShapedMembershipFunction ... 3206
A ... 3208
B ... 3208
GetValue ... 3209
CSigmoidalMembershipFunction ... 3210
A ... 3212
C ... 3212
GetValue ... 3213
CTrapezoidMembershipFunction ... 3214
X1 ... 3216
X2 ... 3216
X3 ... 3217
X4 ... 3217
GetValue ... 3218
CTriangularMembershipFunction ... 3219
X1 ... 3221
X2 ... 3221
X3 ... 3222
ToNormalMF ... 3222
GetValue ... 3222
CZ_ShapedMembershipFunction ... 3224
A ... 3226
B ... 3226
GetValue ... 3227
IMembershipFunction ... 3228
GetValue ... 3228
Regras para sistemas difusos ... 3229
CMamdaniFuzzyRule ... 3230
Conclusion ... 3230
Weight ... 3231
CSugenoFuzzyRule ... 3232
Conclusion ... 3232
CSingleCondition ... 3234
Not ... 3234
Term ... 3235
Var ... 3235
CConditions ... 3237
ConditionsList ... 3237
Not ... 3238
Op ... 3238
CGenericFuzzyRule ... 3240
Conclusion ... 3240
Condition ... 3241
CreateCondition ... 3241
Variáveis para sistemas difusos ... 3243
CFuzzyVariable ... 3244
AddTerm ... 3245
GetTermByName ... 3245
Max ... 3245
Min ... 3246
Terms ... 3246

© 2000-2024, MetaQuotes Ltd.

Values ... 3247
CSugenoVariable ... 3248
Functions ... 3248
GetFuncByName ... 3249
Values ... 3249
Termos difusos ... 3250
MembershipFunction ... 3251
Sistemas difusos ... 3252
Sistema Mamdani ... 3253
AggregationMethod ... 3253
Calculate ... 3254
DefuzzificationMethod ... 3254
EmptyRule ... 3254
ImplicationMethod ... 3254
Output ... 3255
OutputByName ... 3255
ParseRule ... 3255
Rules ... 3255
Sistema Sugeno ... 3257
Calculate ... 3257
CreateSugenoFunction ... 3258
EmptyRule ... 3259
Output ... 3259
OutputByName ... 3259
ParseRule ... 3259
Rules ... 3260
OpenCL ... 3261
BufferCreate ... 3263
BufferFree ... 3264
BufferFromArray ... 3265
BufferRead ... 3266
BufferWrite ... 3267
Execute ... 3268
GetContext ... 3269
GetKernel ... 3270
GetKernelName ... 3271
GetProgram ... 3272
Initialize ... 3273
KernelCreate ... 3274
KernelFree ... 3275
SetArgument ... 3276
SetArgumentBuffer ... 3277
SetArgumentLocalMemory ... 3278
SetBuffersCount ... 3279
SetKernelsCount ... 3280
Shutdown ... 3281
SupportDouble ... 3282
Classe base CObject ... 3283
Prev ... 3284
Prev ... 3285
Next ... 3286
Next ... 3287
Compare ... 3288
Save ... 3290
Load ... 3292
Type ... 3294
Coleções de dados ... 3295
CArray ... 3296

© 2000-2024, MetaQuotes Ltd.

Step ... 3298
Step ... 3299
Total ... 3300
Available ... 3301
Max ... 3302
IsSorted ... 3303
SortMode ... 3304
Clear ... 3305
Sort ... 3306
Save ... 3307
Load ... 3308
CArrayChar ... 3309
Reserve ... 3312
Resize ... 3313
Shutdown ... 3314
Add ... 3315
AddArray ... 3316
AddArray ... 3317
Insert ... 3319
InsertArray ... 3320
InsertArray ... 3321
AssignArray ... 3323
AssignArray ... 3324
Update ... 3326
Shift ... 3327
Delete ... 3328
DeleteRange ... 3329
At ... 3330
CompareArray ... 3332
CompareArray ... 3333
InsertSort ... 3334
Search ... 3335
SearchGreat ... 3336
SearchLess ... 3337
SearchGreatOrEqual ... 3338
SearchLessOrEqual ... 3339
SearchFirst ... 3340
SearchLast ... 3341
SearchLinear ... 3342
Save ... 3343
Load ... 3344
Type ... 3346
CArrayShort ... 3347
Reserve ... 3349
Resize ... 3350
Shutdown ... 3351
Add ... 3352
AddArray ... 3353
AddArray ... 3354
Insert ... 3356
InsertArray ... 3357
InsertArray ... 3358
AssignArray ... 3360
AssignArray ... 3361
Update ... 3363
Shift ... 3364
Delete ... 3365
DeleteRange ... 3366

© 2000-2024, MetaQuotes Ltd.

At ... 3367
CompareArray ... 3369
CompareArray ... 3370
InsertSort ... 3371
Search ... 3372
SearchGreat ... 3373
SearchLess ... 3374
SearchGreatOrEqual ... 3375
SearchLessOrEqual ... 3376
SearchFirst ... 3377
SearchLast ... 3378
SearchLinear ... 3379
Save ... 3380
Load ... 3382
Type ... 3384
CArrayInt ... 3385
Reserve ... 3388
Resize ... 3389
Shutdown ... 3390
Add ... 3391
AddArray ... 3392
AddArray ... 3393
Insert ... 3395
InsertArray ... 3396
InsertArray ... 3397
AssignArray ... 3399
AssignArray ... 3400
Update ... 3402
Shift ... 3403
Delete ... 3404
DeleteRange ... 3405
At ... 3406
CompareArray ... 3408
CompareArray ... 3409
InsertSort ... 3410
Search ... 3411
SearchGreat ... 3412
SearchLess ... 3413
SearchGreatOrEqual ... 3414
SearchLessOrEqual ... 3415
SearchFirst ... 3416
SearchLast ... 3417
SearchLinear ... 3418
Save ... 3419
Load ... 3421
Type ... 3423
CArrayLong ... 3424
Reserve ... 3427
Resize ... 3428
Shutdown ... 3429
Add ... 3430
AddArray ... 3431
AddArray ... 3432
Insert ... 3434
InsertArray ... 3435
InsertArray ... 3436
AssignArray ... 3438
AssignArray ... 3439

© 2000-2024, MetaQuotes Ltd.

Update ... 3441
Shift ... 3442
Delete ... 3443
DeleteRange ... 3444
At ... 3445
CompareArray ... 3447
CompareArray ... 3448
InsertSort ... 3449
Search ... 3450
SearchGreat ... 3451
SearchLess ... 3452
SearchGreatOrEqual ... 3453
SearchLessOrEqual ... 3454
SearchFirst ... 3455
SearchLast ... 3456
SearchLinear ... 3457
Save ... 3458
Load ... 3460
Type ... 3462
CArrayFloat ... 3463
Delta ... 3466
Reserve ... 3467
Resize ... 3468
Shutdown ... 3469
Add ... 3470
AddArray ... 3471
AddArray ... 3472
Insert ... 3474
InsertArray ... 3475
InsertArray ... 3476
AssignArray ... 3478
AssignArray ... 3479
Update ... 3481
Shift ... 3482
Delete ... 3483
DeleteRange ... 3484
At ... 3485
CompareArray ... 3487
CompareArray ... 3488
InsertSort ... 3489
Search ... 3490
SearchGreat ... 3491
SearchLess ... 3492
SearchGreatOrEqual ... 3493
SearchLessOrEqual ... 3494
SearchFirst ... 3495
SearchLast ... 3496
SearchLinear ... 3497
Save ... 3498
Load ... 3500
Type ... 3502
CArrayDouble ... 3503
Delta ... 3506
Reserve ... 3507
Resize ... 3508
Shutdown ... 3509
Add ... 3510
AddArray ... 3511

© 2000-2024, MetaQuotes Ltd.

AddArray ... 3512
Insert ... 3514
InsertArray ... 3515
InsertArray ... 3516
AssignArray ... 3518
AssignArray ... 3519
Update ... 3521
Shift ... 3522
Delete ... 3523
DeleteRange ... 3524
At ... 3525
CompareArray ... 3527
CompareArray ... 3528
Minimum ... 3529
Maximum ... 3530
InsertSort ... 3531
Search ... 3532
SearchGreat ... 3533
SearchLess ... 3534
SearchGreatOrEqual ... 3535
SearchLessOrEqual ... 3536
SearchFirst ... 3537
SearchLast ... 3538
SearchLinear ... 3539
Save ... 3540
Load ... 3542
Type ... 3544
CArrayString ... 3545
Reserve ... 3547
Resize ... 3548
Shutdown ... 3549
Add ... 3550
AddArray ... 3551
AddArray ... 3552
Insert ... 3554
InsertArray ... 3555
InsertArray ... 3556
AssignArray ... 3558
AssignArray ... 3559
Update ... 3561
Shift ... 3562
Delete ... 3563
DeleteRange ... 3564
At ... 3565
CompareArray ... 3567
CompareArray ... 3568
InsertSort ... 3569
Search ... 3570
SearchGreat ... 3571
SearchLess ... 3572
SearchGreatOrEqual ... 3573
SearchLessOrEqual ... 3574
SearchFirst ... 3575
SearchLast ... 3576
SearchLinear ... 3577
Save ... 3578
Load ... 3580
Type ... 3582

© 2000-2024, MetaQuotes Ltd.

CArrayObj ... 3583
FreeMode ... 3588
FreeMode ... 3589
Reserve ... 3591
Resize ... 3592
Clear ... 3593
Shutdown ... 3594
CreateElement ... 3595
Add ... 3597
AddArray ... 3598
Insert ... 3601
InsertArray ... 3603
AssignArray ... 3605
Update ... 3607
Shift ... 3608
Detach ... 3609
Delete ... 3611
DeleteRange ... 3612
At ... 3613
CompareArray ... 3614
InsertSort ... 3615
Search ... 3616
SearchGreat ... 3617
SearchLess ... 3618
SearchGreatOrEqual ... 3619
SearchLessOrEqual ... 3620
SearchFirst ... 3621
SearchLast ... 3622
Save ... 3623
Load ... 3624
Type ... 3626
CList ... 3627
FreeMode ... 3629
FreeMode ... 3630
Total ... 3632
IsSorted ... 3633
SortMode ... 3634
CreateElement ... 3635
Add ... 3636
Insert ... 3637
DetachCurrent ... 3639
DeleteCurrent ... 3640
Delete ... 3641
Clear ... 3642
IndexOf ... 3643
GetNodeAtIndex ... 3644
GetFirstNode ... 3645
GetPrevNode ... 3646
GetCurrentNode ... 3647
GetNextNode ... 3648
GetLastNode ... 3649
Sort ... 3650
MoveToIndex ... 3651
Exchange ... 3652
CompareList ... 3653
Search ... 3654
Save ... 3655
Load ... 3657

© 2000-2024, MetaQuotes Ltd.

Type ... 3659
CTreeNode ... 3660
Owner ... 3665
Left ... 3666
Right ... 3667
Balance ... 3668
BalanceL ... 3669
BalanceR ... 3670
CreateSample ... 3671
RefreshBalance ... 3672
GetNext ... 3673
SaveNode ... 3674
LoadNode ... 3675
Type ... 3676
CTree ... 3677
Root ... 3683
CreateElement ... 3684
Insert ... 3685
Detach ... 3686
Delete ... 3687
Clear ... 3688
Find ... 3689
Save ... 3690
Load ... 3691
Type ... 3692
Coleções de dados genéricas ... 3693
ICollection<T> ... 3696
Add ... 3697
Count ... 3698
Contains ... 3699
CopyTo ... 3700
Clear ... 3701
Remove ... 3702
IEqualityComparable<T> ... 3703
Equals ... 3704
HashCode ... 3705
IComparable<T> ... 3706
Compare ... 3707
IComparer<T> ... 3708
Compare ... 3709
IEqualityComparer<T> ... 3710
Equals ... 3711
HashCode ... 3712
IList<T> ... 3713
TryGetValue ... 3714
TrySetValue ... 3715
Insert ... 3716
IndexOf ... 3717
LastIndexOf ... 3718
RemoveAt ... 3719
IMap<TKey,TValue> ... 3720
Add ... 3721
Contains ... 3722
Remove ... 3723
TryGetValue ... 3724
TrySetValue ... 3725
CopyTo ... 3726
ISet<T> ... 3727

© 2000-2024, MetaQuotes Ltd.

ExceptWith ... 3729
IntersectWith ... 3730
SymmetricExceptWith ... 3731
UnionWith ... 3732
IsProperSubsetOf ... 3733
IsProperSupersetOf ... 3734
IsSubsetOf ... 3735
IsSupersetOf ... 3736
Overlaps ... 3737
SetEquals ... 3738
CDefaultComparer<T> ... 3739
Compare ... 3740
CDefaultEqualityComparer<T> ... 3741
Equals ... 3742
HashCode ... 3743
CRedBlackTreeNode<T> ... 3744
Value ... 3745
Parent ... 3746
Left ... 3747
Right ... 3748
Color ... 3749
IsLeaf ... 3750
CreateEmptyNode ... 3751
CLinkedListNode<T> ... 3752
List ... 3753
Next ... 3754
Previous ... 3755
Value ... 3756
CKeyValuePair<TKey,TValue> ... 3757
Key ... 3758
Value ... 3759
Clone ... 3760
Compare ... 3761
Equals ... 3762
HashCode ... 3763
CArrayList<T> ... 3764
Capacity ... 3766
Count ... 3767
Contains ... 3768
TrimExcess ... 3769
TryGetValue ... 3770
TrySetValue ... 3771
Add ... 3772
AddRange ... 3773
Insert ... 3774
InsertRange ... 3775
CopyTo ... 3776
BinarySearch ... 3777
IndexOf ... 3778
LastIndexOf ... 3779
Clear ... 3780
Remove ... 3781
RemoveAt ... 3782
RemoveRange ... 3783
Reverse ... 3784
Sort ... 3785
CHashMap<TKey,TValue> ... 3786
Add ... 3788

© 2000-2024, MetaQuotes Ltd.

Count ... 3789
Comparer ... 3790
Contains ... 3791
ContainsKey ... 3792
ContainsValue ... 3793
CopyTo ... 3794
Clear ... 3795
Remove ... 3796
TryGetValue ... 3797
TrySetValue ... 3798
CHashSet<T> ... 3799
Add ... 3801
Count ... 3802
Contains ... 3803
Comparer ... 3804
TrimExcess ... 3805
CopyTo ... 3806
Clear ... 3807
Remove ... 3808
ExceptWith ... 3809
IntersectWith ... 3810
SymmetricExceptWith ... 3811
UnionWith ... 3812
IsProperSubsetOf ... 3813
IsProperSupersetOf ... 3814
IsSubsetOf ... 3815
IsSupersetOf ... 3816
Overlaps ... 3817
SetEquals ... 3818
CLinkedList<T> ... 3819
Add ... 3821
AddAfter ... 3822
AddBefore ... 3823
AddFirst ... 3824
AddLast ... 3825
Count ... 3826
Head ... 3827
First ... 3828
Last ... 3829
Contains ... 3830
CopyTo ... 3831
Clear ... 3832
Remove ... 3833
RemoveFirst ... 3834
RemoveLast ... 3835
Find ... 3836
FindLast ... 3837
CQueue<T> ... 3838
Add ... 3839
Enqueue ... 3840
Count ... 3841
Contains ... 3842
TrimExcess ... 3843
CopyTo ... 3844
Clear ... 3845
Remove ... 3846
Dequeue ... 3847
Peek ... 3848

© 2000-2024, MetaQuotes Ltd.

CRedBlackTree<T> ... 3849
Add ... 3851
Count ... 3852
Root ... 3853
Contains ... 3854
Comparer ... 3855
TryGetMin ... 3856
TryGetMax ... 3857
CopyTo ... 3858
Clear ... 3859
Remove ... 3860
RemoveMin ... 3861
RemoveMax ... 3862
Find ... 3863
FindMin ... 3864
FindMax ... 3865
CSortedMap<TKey, TValue> ... 3866
Add ... 3868
Count ... 3869
Comparer ... 3870
Contains ... 3871
ContainsKey ... 3872
ContainsValue ... 3873
CopyTo ... 3874
Clear ... 3875
Remove ... 3876
TryGetValue ... 3877
TrySetValue ... 3878
CSortedSet<T> ... 3879
Add ... 3881
Count ... 3882
Contains ... 3883
Comparer ... 3884
TryGetMin ... 3885
TryGetMax ... 3886
CopyTo ... 3887
Clear ... 3888
Remove ... 3889
ExceptWith ... 3890
IntersectWith ... 3891
SymmetricExceptWith ... 3892
UnionWith ... 3893
IsProperSubsetOf ... 3894
IsProperSupersetOf ... 3895
IsSubsetOf ... 3896
IsSupersetOf ... 3897
Overlaps ... 3898
SetEquals ... 3899
GetViewBetween ... 3900
GetReverse ... 3901
CStack<T> ... 3902
Add ... 3903
Count ... 3904
Contains ... 3905
TrimExcess ... 3906
CopyTo ... 3907
Clear ... 3908
Remove ... 3909

© 2000-2024, MetaQuotes Ltd.

Push ... 3910
Peek ... 3911
Pop ... 3912
ArrayBinarySearch<T> ... 3913
ArrayIndexOf<T> ... 3914
ArrayLastIndexOf<T> ... 3915
ArrayReverse<T> ... 3916
Compare ... 3917
Equals<T> ... 3920
GetHashCode ... 3921
Arquivos ... 3924
CFile ... 3925
Handle ... 3927
Filename ... 3928
Flags ... 3929
SetUnicode ... 3930
SetCommon ... 3931
Open ... 3932
Close ... 3933
Delete ... 3934
IsExist ... 3935
Copy ... 3936
Move ... 3937
Size ... 3938
Tell ... 3939
Seek ... 3940
Flush ... 3941
IsEnding ... 3942
IsLineEnding ... 3943
FolderCreate ... 3944
FolderDelete ... 3945
FolderClean ... 3946
FileFindFirst ... 3947
FileFindNext ... 3948
FileFindClose ... 3949
CFileBin ... 3950
Open ... 3952
WriteChar ... 3953
WriteShort ... 3954
WriteInteger ... 3955
WriteLong ... 3956
WriteFloat ... 3957
WriteDouble ... 3958
WriteString ... 3959
WriteCharArray ... 3960
WriteShortArray ... 3961
WriteIntegerArray ... 3962
WriteLongArray ... 3963
WriteFloatArray ... 3964
WriteDoubleArray ... 3965
WriteObject ... 3966
ReadChar ... 3967
ReadShort ... 3968
ReadInteger ... 3969
ReadLong ... 3970
ReadFloat ... 3971
ReadDouble ... 3972
ReadString ... 3973

© 2000-2024, MetaQuotes Ltd.

ReadCharArray ... 3974
ReadShortArray ... 3975
ReadIntegerArray ... 3976
ReadLongArray ... 3977
ReadFloatArray ... 3978
ReadDoubleArray ... 3979
ReadObject ... 3980
CFileTxt ... 3981
Open ... 3982
WriteString ... 3983
ReadString ... 3984
Cadeias de caracteres ... 3985
CString ... 3986
Str ... 3988
Len ... 3989
Copy ... 3990
Fill ... 3991
Assign ... 3992
Append ... 3993
Insert ... 3994
Compare ... 3995
CompareNoCase ... 3996
Left ... 3997
Right ... 3998
Mid ... 3999
Trim ... 4000
TrimLeft ... 4001
TrimRight ... 4002
Clear ... 4003
ToUpper ... 4004
ToLower ... 4005
Reverse ... 4006
Find ... 4007
FindRev ... 4008
Remove ... 4009
Replace ... 4010
Objetos gráficos ... 4011
CChartObject ... 4012
ChartId ... 4015
Window ... 4016
Name ... 4017
NumPoints ... 4018
Attach ... 4019
SetPoint ... 4020
Delete ... 4021
Detach ... 4022
ShiftObject ... 4023
ShiftPoint ... 4024
Time ... 4025
Price ... 4027
Color ... 4029
Style ... 4030
Width ... 4031
Background ... 4032
Selected ... 4033
Selectable ... 4034
Description ... 4035
Tooltip ... 4036

© 2000-2024, MetaQuotes Ltd.

Timeframes ... 4037
Z_Order ... 4038
CreateTime ... 4039
LevelsCount ... 4040
LevelColor ... 4041
LevelStyle ... 4043
LevelWidth ... 4045
LevelValue ... 4047
LevelDescription ... 4049
GetInteger ... 4051
SetInteger ... 4053
GetDouble ... 4055
SetDouble ... 4057
GetString ... 4059
SetString ... 4061
Save ... 4063
Load ... 4064
Type ... 4065
Objetos de Linhas ... 4066
CChartObjectVLine ... 4067
Create ... 4068
Type ... 4069
CChartObjectHLine ... 4070
Create ... 4071
Type ... 4072
CChartObjectTrend ... 4073
Create ... 4075
RayLeft ... 4076
RayRight ... 4077
Save ... 4078
Load ... 4079
Type ... 4080
CChartObjectTrendByAngle ... 4081
Create ... 4083
Angle ... 4084
Type ... 4085
CChartObjectCycles ... 4086
Create ... 4087
Type ... 4088
Objetos de Canais ... 4089
CChartObjectChannel ... 4090
Create ... 4092
Type ... 4093
CChartObjectRegression ... 4094
Create ... 4096
Type ... 4097
CChartObjectStdDevChannel ... 4098
Create ... 4100
Deviations ... 4101
Save ... 4102
Load ... 4103
Type ... 4104
CChartObjectPitchfork ... 4105
Create ... 4107
Type ... 4108
Ferramentas Gann ... 4109
CChartObjectGannLine ... 4110
Create ... 4112

© 2000-2024, MetaQuotes Ltd.

PipsPerBar ... 4113
Save ... 4114
Load ... 4115
Type ... 4116
CChartObjectGannFan ... 4117
Create ... 4119
PipsPerBar ... 4120
Downtrend ... 4121
Save ... 4122
Load ... 4123
Type ... 4124
CChartObjectGannGrid ... 4125
Create ... 4127
PipsPerBar ... 4128
Downtrend ... 4129
Save ... 4130
Load ... 4131
Type ... 4132
Ferramentas Fibonacci ... 4133
CChartObjectFibo ... 4134
Create ... 4136
Type ... 4137
CChartObjectFiboTimes ... 4138
Create ... 4139
Type ... 4140
CChartObjectFiboFan ... 4141
Create ... 4142
Type ... 4143
CChartObjectFiboArc ... 4144
Create ... 4146
Scale ... 4147
Ellipse ... 4148
Save ... 4149
Load ... 4150
Type ... 4151
CChartObjectFiboChannel ... 4152
Create ... 4154
Type ... 4155
CChartObjectFiboExpansion ... 4156
Create ... 4158
Type ... 4159
Ferramentas Elliott ... 4160
CChartObjectElliottWave3 ... 4161
Create ... 4163
Degree ... 4164
Lines ... 4165
Save ... 4166
Load ... 4167
Type ... 4168
CChartObjectElliottWave5 ... 4169
Create ... 4171
Type ... 4173
Objetos de Formas ... 4174
CChartObjectRectangle ... 4175
Create ... 4176
Type ... 4177
CChartObjectTriangle ... 4178
Create ... 4179

© 2000-2024, MetaQuotes Ltd.

Type ... 4180
CChartObjectEllipse ... 4181
Create ... 4182
Type ... 4183
Objetos de Setas ... 4184
CChartObjectArrow ... 4185
Create ... 4187
ArrowCode ... 4189
Anchor ... 4191
Save ... 4193
Load ... 4194
Type ... 4195
Arrows with fixed code ... 4196
Create ... 4198
ArrowCode ... 4200
Type ... 4201
Objetos de Controles ... 4202
CChartObjectText ... 4203
Create ... 4205
Angle ... 4206
Font ... 4207
FontSize ... 4208
Anchor ... 4209
Save ... 4210
Load ... 4211
Type ... 4212
CChartObjectLabel ... 4213
Create ... 4215
X_Distance ... 4216
Y_Distance ... 4217
X_Size ... 4218
Y_Size ... 4219
Corner ... 4220
Time ... 4221
Price ... 4222
Save ... 4223
Load ... 4224
Type ... 4225
CChartObjectEdit ... 4226
Create ... 4228
TextAlign ... 4229
X_Size ... 4230
Y_Size ... 4231
BackColor ... 4232
BorderColor ... 4233
ReadOnly ... 4234
Angle ... 4235
Save ... 4236
Load ... 4237
Type ... 4238
CChartObjectButton ... 4239
State ... 4241
Save ... 4242
Load ... 4243
Type ... 4244
CChartObjectSubChart ... 4245
Create ... 4247
X_Distance ... 4248

© 2000-2024, MetaQuotes Ltd.

Y_Distance ... 4249
Corner ... 4250
X_Size ... 4251
Y_Size ... 4252
Symbol ... 4253
Period ... 4254
Scale ... 4255
DateScale ... 4256
PriceScale ... 4257
Time ... 4258
Price ... 4259
Save ... 4260
Load ... 4261
Type ... 4262
CChartObjectBitmap ... 4263
Create ... 4265
BmpFile ... 4266
X_Offset ... 4267
Y_Offset ... 4268
Save ... 4269
Load ... 4270
Type ... 4271
CChartObjectBmpLabel ... 4272
Create ... 4274
X_Distance ... 4275
Y_Distance ... 4276
X_Offset ... 4277
Y_Offset ... 4278
Corner ... 4279
X_Size ... 4280
Y_Size ... 4281
BmpFileOn ... 4282
BmpFileOff ... 4283
State ... 4284
Time ... 4285
Price ... 4286
Save ... 4287
Load ... 4288
Type ... 4289
CChartObjectRectLabel ... 4290
Create ... 4292
X_Size ... 4293
Y_Size ... 4294
BackColor ... 4295
Angle ... 4296
BorderType ... 4297
Save ... 4298
Load ... 4299
Type ... 4300
Gráficos personalizados ... 4301
CCanvas ... 4302
Attach ... 4306
Arc ... 4307
Pie ... 4311
FillPolygon ... 4315
FillEllipse ... 4316
GetDefaultColor ... 4317
ChartObjectName ... 4318

© 2000-2024, MetaQuotes Ltd.

Circle ... 4319
CircleAA ... 4320
CircleWu ... 4321
Create ... 4322
CreateBitmap ... 4323
CreateBitmapLabel ... 4325
Destroy ... 4327
Ellipse ... 4328
EllipseAA ... 4329
EllipseWu ... 4330
Erase ... 4331
Fill ... 4332
FillCircle ... 4333
FillRectangle ... 4334
FillTriangle ... 4335
FontAngleGet ... 4336
FontAngleSet ... 4337
FontFlagsGet ... 4338
FontFlagsSet ... 4339
FontGet ... 4340
FontNameGet ... 4341
FontNameSet ... 4342
FontSet ... 4343
FontSizeGet ... 4344
FontSizeSet ... 4345
Height ... 4346
Line ... 4347
LineAA ... 4348
LineWu ... 4349
LineHorizontal ... 4350
LineVertical ... 4351
LineStyleSet ... 4352
LineThick ... 4353
LineThickVertical ... 4355
LineThickHorizontal ... 4356
LoadFromFile ... 4357
PixelGet ... 4358
PixelSet ... 4359
PixelSetAA ... 4360
Polygon ... 4361
PolygonAA ... 4362
PolygonWu ... 4363
PolygonThick ... 4364
PolygonSmooth ... 4365
Polyline ... 4366
PolylineSmooth ... 4367
PolylineThick ... 4368
PolylineWu ... 4369
PolylineAA ... 4370
Rectangle ... 4371
Resize ... 4372
ResourceName ... 4373
TextHeight ... 4374
TextOut ... 4375
TextSize ... 4376
TextWidth ... 4377
TransparentLevelSet ... 4378
Triangle ... 4379

© 2000-2024, MetaQuotes Ltd.

TriangleAA ... 4380
TriangleWu ... 4381
Update ... 4382
Width ... 4383
CChartCanvas ... 4384
ColorBackground ... 4388
ColorBorder ... 4389
ColorText ... 4390
ColorGrid ... 4391
MaxData ... 4392
MaxDescrLen ... 4393
ShowFlags ... 4394
IsShowLegend ... 4395
IsShowScaleLeft ... 4396
IsShowScaleRight ... 4397
IsShowScaleTop ... 4398
IsShowScaleBottom ... 4399
IsShowGrid ... 4400
IsShowDescriptors ... 4401
IsShowPercent ... 4402
VScaleMin ... 4403
VScaleMax ... 4404
NumGrid ... 4405
DataOffset ... 4406
DataTotal ... 4407
DrawDescriptors ... 4408
DrawData ... 4409
Create ... 4410
AllowedShowFlags ... 4411
ShowLegend ... 4412
ShowScaleLeft ... 4413
ShowScaleRight ... 4414
ShowScaleTop ... 4415
ShowScaleBottom ... 4416
ShowGrid ... 4417
ShowDescriptors ... 4418
ShowValue ... 4419
ShowPercent ... 4420
LegendAlignment ... 4421
Accumulative ... 4422
VScaleParams ... 4423
DescriptorUpdate ... 4424
ColorUpdate ... 4425
ValuesCheck ... 4426
Redraw ... 4427
DrawBackground ... 4428
DrawLegend ... 4429
DrawLegendVertical ... 4430
DrawLegendHorizontal ... 4431
CalcScales ... 4432
DrawScales ... 4433
DrawScaleLeft ... 4434
DrawScaleRight ... 4435
DrawScaleTop ... 4436
DrawScaleBottom ... 4437
DrawGrid ... 4438
DrawChart ... 4439
CHistogramChart ... 4440

© 2000-2024, MetaQuotes Ltd.

Gradient ... 4445
BarGap ... 4446
BarMinSize ... 4447
BarBorder ... 4448
Create ... 4449
SeriesAdd ... 4450
SeriesInsert ... 4451
SeriesUpdate ... 4452
SeriesDelete ... 4453
ValueUpdate ... 4454
DrawData ... 4455
DrawBar ... 4456
GradientBrush ... 4457
CLineChart ... 4458
Filled ... 4462
Create ... 4463
SeriesAdd ... 4464
SeriesInsert ... 4465
SeriesUpdate ... 4466
SeriesDelete ... 4467
ValueUpdate ... 4468
DrawChart ... 4469
DrawData ... 4470
CalcArea ... 4471
CPieChart ... 4472
Create ... 4476
SeriesSet ... 4477
ValueAdd ... 4478
ValueInsert ... 4479
ValueUpdate ... 4480
ValueDelete ... 4481
DrawChart ... 4482
DrawPie ... 4483
LabelMake ... 4484
Gráfico 3D ... 4485
CCanvas3D ... 4486
AmbientColorGet ... 4488
AmbientColorSet ... 4489
Attach ... 4490
Create ... 4491
Destroy ... 4492
DXContext ... 4493
DXDispatcher ... 4494
InputScene ... 4495
LightColorGet ... 4496
LightColorSet ... 4497
LightDirectionGet ... 4498
LightDirectionSet ... 4499
ObjectAdd ... 4500
ProjectionMatrixGet ... 4501
ProjectionMatrixSet ... 4502
Render ... 4503
RenderBegin ... 4504
RenderEnd ... 4505
ViewMatrixGet ... 4506
ViewMatrixSet ... 4507
ViewPositionSet ... 4508
ViewRotationSet ... 4509

© 2000-2024, MetaQuotes Ltd.

ViewTargetSet ... 4510
ViewUpDirectionSet ... 4511
Gráficos de preços ... 4512
ChartID ... 4517
Mode ... 4518
Foreground ... 4519
Shift ... 4520
ShiftSize ... 4521
AutoScroll ... 4522
Scale ... 4523
ScaleFix ... 4524
ScaleFix_11 ... 4525
FixedMax ... 4526
FixedMin ... 4527
PointsPerBar ... 4528
ScalePPB ... 4529
ShowOHLC ... 4530
ShowLineBid ... 4531
ShowLineAsk ... 4532
ShowLastLine ... 4533
ShowPeriodSep ... 4534
ShowGrid ... 4535
ShowVolumes ... 4536
ShowObjectDescr ... 4537
ShowDateScale ... 4538
ShowPriceScale ... 4539
ColorBackground ... 4540
ColorForeground ... 4541
ColorGrid ... 4542
ColorBarUp ... 4543
ColorBarDown ... 4544
ColorCandleBull ... 4545
ColorCandleBear ... 4546
ColorChartLine ... 4547
ColorVolumes ... 4548
ColorLineBid ... 4549
ColorLineAsk ... 4550
ColorLineLast ... 4551
ColorStopLevels ... 4552
VisibleBars ... 4553
WindowsTotal ... 4554
WindowIsVisible ... 4555
WindowHandle ... 4556
FirstVisibleBar ... 4557
WidthInBars ... 4558
WidthInPixels ... 4559
HeightInPixels ... 4560
PriceMin ... 4561
PriceMax ... 4562
Attach ... 4563
FirstChart ... 4564
NextChart ... 4565
Open ... 4566
Detach ... 4567
Close ... 4568
BringToTop ... 4569
EventObjectCreate ... 4570
EventObjectDelete ... 4571

© 2000-2024, MetaQuotes Ltd.

IndicatorAdd ... 4572
IndicatorDelete ... 4573
IndicatorsTotal ... 4574
IndicatorName ... 4575
Navigate ... 4576
Symbol ... 4577
Period ... 4578
Redraw ... 4579
GetInteger ... 4580
SetInteger ... 4581
GetDouble ... 4582
SetDouble ... 4583
GetString ... 4584
SetString ... 4585
SetSymbolPeriod ... 4586
ApplyTemplate ... 4587
ScreenShot ... 4588
WindowOnDropped ... 4589
PriceOnDropped ... 4590
TimeOnDropped ... 4591
XOnDropped ... 4592
YOnDropped ... 4593
Save ... 4594
Load ... 4595
Type ... 4596
Gráficos científicos ... 4597
GraphPlot ... 4598
CAxis ... 4602
AutoScale ... 4604
Min ... 4605
Max ... 4606
Step ... 4607
Name ... 4608
Color ... 4609
ValuesSize ... 4610
ValuesWidth ... 4611
ValuesFormat ... 4612
ValuesDateTimeMode ... 4613
ValuesFunctionFormat ... 4614
ValuesFunctionFormatCBData ... 4616
NameSize ... 4617
ZeroLever ... 4618
DefaultStep ... 4619
MaxLabels ... 4620
MinGrace ... 4621
MaxGrace ... 4622
SelectAxisScale ... 4623
CColorGenerator ... 4624
Next ... 4625
Reset ... 4626
CCurve ... 4627
Type ... 4629
Name ... 4630
Color ... 4631
XMax ... 4632
XMin ... 4633
YMax ... 4634
YMin ... 4635

© 2000-2024, MetaQuotes Ltd.

Size ... 4636
PointsSize ... 4637
PointsFill ... 4638
PointsColor ... 4639
GetX ... 4640
GetY ... 4641
LinesStyle ... 4642
LinesIsSmooth ... 4643
LinesSmoothTension ... 4644
LinesSmoothStep ... 4645
LinesEndStyle ... 4646
LinesWidth ... 4647
HistogramWidth ... 4649
CustomPlotCBData ... 4650
CustomPlotFunction ... 4651
PointsType ... 4655
StepsDimension ... 4656
TrendLineCoefficients ... 4657
TrendLineColor ... 4658
TrendLineVisible ... 4659
Update ... 4661
Visible ... 4663
CGraphic ... 4664
Create ... 4667
Destroy ... 4668
Update ... 4669
ChartObjectName ... 4670
ResourceName ... 4671
XAxis ... 4672
YAxis ... 4673
GapSize ... 4674
BackgroundColor ... 4675
BackgroundMain ... 4676
BackgroundMainSize ... 4677
BackgroundMainColor ... 4678
BackgroundSub ... 4679
BackgroundSubSize ... 4680
BackgroundSubColor ... 4681
GridLineColor ... 4682
GridBackgroundColor ... 4683
GridCircleRadius ... 4684
GridCircleColor ... 4685
GridHasCircle ... 4686
HistoryNameWidth ... 4687
HistoryNameSize ... 4688
HistorySymbolSize ... 4689
TextAdd ... 4690
LineAdd ... 4691
CurveAdd ... 4692
CurvePlot ... 4695
CurvePlotAll ... 4696
CurveGetByIndex ... 4697
CurveGetByName ... 4698
CurveRemoveByIndex ... 4699
CurveRemoveByName ... 4700
CurvesTotal ... 4701
MarksToAxisAdd ... 4702
MajorMarkSize ... 4703

© 2000-2024, MetaQuotes Ltd.

FontSet ... 4704
FontGet ... 4705
GridAxisLineColor ... 4706
Attach ... 4707
CalculateMaxMinValues ... 4708
Height ... 4709
IndentDown ... 4710
IndentLeft ... 4711
IndentRight ... 4712
IndentUp ... 4713
Redraw ... 4714
ResetParameters ... 4715
SetDefaultParameters ... 4716
ScaleX ... 4717
ScaleY ... 4718
Width ... 4719
Indicadores ... 4720
Classes Base ... 4721
CSpreadBuffer ... 4722
Size ... 4724
SetSymbolPeriod ... 4725
At ... 4726
Refresh ... 4727
RefreshCurrent ... 4728
CTimeBuffer ... 4729
Size ... 4731
SetSymbolPeriod ... 4732
At ... 4733
Refresh ... 4734
RefreshCurrent ... 4735
CTickVolumeBuffer ... 4736
Size ... 4738
SetSymbolPeriod ... 4739
At ... 4740
Refresh ... 4741
RefreshCurrent ... 4742
CRealVolumeBuffer ... 4743
Size ... 4745
SetSymbolPeriod ... 4746
At ... 4747
Refresh ... 4748
RefreshCurrent ... 4749
CDoubleBuffer ... 4750
Size ... 4752
SetSymbolPeriod ... 4753
At ... 4754
Refresh ... 4755
RefreshCurrent ... 4756
COpenBuffer ... 4757
Refresh ... 4758
RefreshCurrent ... 4759
CHighBuffer ... 4760
Refresh ... 4761
RefreshCurrent ... 4762
CLowBuffer ... 4763
Refresh ... 4764
RefreshCurrent ... 4765
CCloseBuffer ... 4766

© 2000-2024, MetaQuotes Ltd.

Refresh ... 4767
RefreshCurrent ... 4768
CIndicatorBuffer ... 4769
Offset ... 4771
Name ... 4772
At ... 4773
Refresh ... 4774
RefreshCurrent ... 4775
CSeries ... 4776
Name ... 4778
BuffersTotal ... 4779
Timeframe ... 4780
Symbol ... 4781
Period ... 4782
RefreshCurrent ... 4783
BufferSize ... 4784
BufferResize ... 4785
Refresh ... 4786
PeriodDescription ... 4787
CPriceSeries ... 4788
BufferResize ... 4790
GetData ... 4791
Refresh ... 4792
MinIndex ... 4793
MinValue ... 4794
MaxIndex ... 4795
MaxValue ... 4796
CIndicator ... 4797
Handle ... 4800
Status ... 4801
FullRelease ... 4802
Create ... 4803
BufferResize ... 4804
BarsCalculated ... 4805
GetData ... 4806
Refresh ... 4809
Minimum ... 4810
MinValue ... 4811
Maximum ... 4812
MaxValue ... 4813
MethodDescription ... 4814
PriceDescription ... 4815
VolumeDescription ... 4816
AddToChart ... 4817
DeleteFromChart ... 4818
CIndicators ... 4819
Create ... 4820
Refresh ... 4821
Classes de timesseries ... 4822
CiSpread ... 4823
Create ... 4825
BufferResize ... 4826
GetData ... 4827
Refresh ... 4829
CiTime ... 4830
Create ... 4832
BufferResize ... 4833
GetData ... 4834

© 2000-2024, MetaQuotes Ltd.

Refresh ... 4836
CiTickVolume ... 4837
Create ... 4839
BufferResize ... 4840
GetData ... 4841
Refresh ... 4843
CiRealVolume ... 4844
Create ... 4846
BufferResize ... 4847
GetData ... 4848
Refresh ... 4850
CiOpen ... 4851
Create ... 4853
GetData ... 4854
CiHigh ... 4856
Create ... 4858
GetData ... 4859
CiLow ... 4861
Create ... 4863
GetData ... 4864
CiClose ... 4866
Create ... 4868
GetData ... 4869
Indicadores de Tendência ... 4871
CiADX ... 4872
MaPeriod ... 4874
Create ... 4875
Main ... 4876
Plus ... 4877
Minus ... 4878
Type ... 4879
CiADXWilder ... 4880
MaPeriod ... 4882
Create ... 4883
Main ... 4884
Plus ... 4885
Minus ... 4886
Type ... 4887
CiBands ... 4888
MaPeriod ... 4890
MaShift ... 4891
Deviation ... 4892
Applied ... 4893
Create ... 4894
Base ... 4895
Upper ... 4896
Lower ... 4897
Type ... 4898
CiEnvelopes ... 4899
MaPeriod ... 4901
MaShift ... 4902
MaMethod ... 4903
Deviation ... 4904
Applied ... 4905
Create ... 4906
Upper ... 4907
Lower ... 4908
Type ... 4909

© 2000-2024, MetaQuotes Ltd.

CiIchimoku 4910
TenkanSenPeriod 4912
KijunSenPeriod 4913
SenkouSpanBPeriod 4914
Create 4915
TenkanSen 4916
KijunSen 4917
SenkouSpanA 4918
SenkouSpanB 4919
ChinkouSpan 4920
Type 4921
CiMA 4922
MaPeriod 4924
MaShift 4925
MaMethod 4926
Applied 4927
Create 4928
Main 4929
Type 4930
CiSAR 4931
SarStep 4933
Maximum 4934
Create 4935
Main 4936
Type 4937
CiStdDev 4938
MaPeriod 4940
MaShift 4941
MaMethod 4942
Applied 4943
Create 4944
Main 4945
Type 4946
CiDEMA 4947
MaPeriod 4949
IndShift 4950
Applied 4951
Create 4952
Main 4953
Type 4954
CiTEMA 4955
MaPeriod 4957
IndShift 4958
Applied 4959
Create 4960
Main 4961
Type 4962
CiFrAMA 4963
MaPeriod 4965
IndShift 4966
Applied 4967
Create 4968
Main 4969
Type 4970
CiAMA 4971
MaPeriod 4973
FastEmaPeriod 4974
SlowEmaPeriod 4975

© 2000-2024, MetaQuotes Ltd.

IndShift ... 4976
Applied ... 4977
Create ... 4978
Main ... 4979
Type ... 4980
CiVIDyA ... 4981
CmoPeriod ... 4983
EmaPeriod ... 4984
IndShift ... 4985
Applied ... 4986
Create ... 4987
Main ... 4988
Type ... 4989
Osciladores ... 4990
CiATR ... 4991
MaPeriod ... 4993
Create ... 4994
Main ... 4995
Type ... 4996
CiBearsPower ... 4997
MaPeriod ... 4999
Create ... 5000
Main ... 5001
Type ... 5002
CiBullsPower ... 5003
MaPeriod ... 5005
Create ... 5006
Main ... 5007
Type ... 5008
CiCCI ... 5009
MaPeriod ... 5011
Applied ... 5012
Create ... 5013
Main ... 5014
Type ... 5015
CiChaikin ... 5016
FastMaPeriod ... 5018
SlowMaPeriod ... 5019
MaMethod ... 5020
Applied ... 5021
Create ... 5022
Main ... 5023
Type ... 5024
CiDeMarker ... 5025
MaPeriod ... 5027
Create ... 5028
Main ... 5029
Type ... 5030
CiForce ... 5031
MaPeriod ... 5033
MaMethod ... 5034
Applied ... 5035
Create ... 5036
Main ... 5037
Type ... 5038
CiMACD ... 5039
FastEmaPeriod ... 5041
SlowEmaPeriod ... 5042

© 2000-2024, MetaQuotes Ltd.

SignalPeriod ... 5043
Applied ... 5044
Create ... 5045
Main ... 5046
Signal ... 5047
Type ... 5048
CiMomentum ... 5049
MaPeriod ... 5051
Applied ... 5052
Create ... 5053
Main ... 5054
Type ... 5055
CiOsMA ... 5056
FastEmaPeriod ... 5058
SlowEmaPeriod ... 5059
SignalPeriod ... 5060
Applied ... 5061
Create ... 5062
Main ... 5063
Type ... 5064
CiRSI ... 5065
MaPeriod ... 5067
Applied ... 5068
Create ... 5069
Main ... 5070
Type ... 5071
CiRVI ... 5072
MaPeriod ... 5074
Create ... 5075
Main ... 5076
Signal ... 5077
Type ... 5078
CiStochastic ... 5079
Kperiod ... 5081
Dperiod ... 5082
Slowing ... 5083
MaMethod ... 5084
PriceField ... 5085
Create ... 5086
Main ... 5087
Signal ... 5088
Type ... 5089
CiTriX ... 5090
MaPeriod ... 5092
Applied ... 5093
Create ... 5094
Main ... 5095
Type ... 5096
CiWPR ... 5097
CalcPeriod ... 5099
Create ... 5100
Main ... 5101
Type ... 5102
Indicadores de Volume ... 5103
CiAD ... 5104
Applied ... 5106
Create ... 5107
Main ... 5108

© 2000-2024, MetaQuotes Ltd.

Type 5109
CiMFI 5110
MaPeriod 5112
Applied 5113
Create 5114
Main 5115
Type 5116
CiOBV 5117
Applied 5119
Create 5120
Main 5121
Type 5122
CiVolumes 5123
Applied 5125
Create 5126
Main 5127
Type 5128
Indicadores Bill Williams 5129
CiAC 5130
Create 5132
Main 5133
Type 5134
CiAlligator 5135
JawPeriod 5137
JawShift 5138
TeethPeriod 5139
TeethShift 5140
LipsPeriod 5141
LipsShift 5142
MaMethod 5143
Applied 5144
Create 5145
Jaw 5146
Teeth 5147
Lips 5148
Type 5149
CiAO 5150
Create 5152
Main 5153
Type 5154
CiFractals 5155
Create 5157
Upper 5158
Lower 5159
Type 5160
CiGator 5161
JawPeriod 5163
JawShift 5164
TeethPeriod 5165
TeethShift 5166
LipsPeriod 5167
LipsShift 5168
MaMethod 5169
Applied 5170
Create 5171
Upper 5172
Lower 5173
Type 5174

© 2000-2024, MetaQuotes Ltd.

CiBWMFI ... 5175
Applied ... 5177
Create ... 5178
Main ... 5179
Type ... 5180
indicadores personalizados ... 5181
NumBuffers ... 5182
NumParams ... 5183
ParamType ... 5184
ParamLong ... 5185
ParamDouble ... 5186
ParamString ... 5187
Type ... 5188
Classes de negociação ... 5189
CAccountInfo ... 5190
Login ... 5192
TradeMode ... 5193
TradeModeDescription ... 5194
Leverage ... 5195
StopoutMode ... 5196
StopoutModeDescription ... 5197
MarginMode ... 5198
MarginModeDescription ... 5199
TradeAllowed ... 5200
TradeExpert ... 5201
LimitOrders ... 5202
Balance ... 5203
Credit ... 5204
Profit ... 5205
Equity ... 5206
Margin ... 5207
FreeMargin ... 5208
MarginLevel ... 5209
MarginCall ... 5210
MarginStopOut ... 5211
Name ... 5212
Server ... 5213
Currency ... 5214
Company ... 5215
InfoInteger ... 5216
InfoDouble ... 5217
InfoString ... 5218
OrderProfitCheck ... 5219
MarginCheck ... 5220
FreeMarginCheck ... 5221
MaxLotCheck ... 5222
CSymbolInfo ... 5223
Refresh ... 5227
RefreshRates ... 5228
Name ... 5229
Select ... 5230
IsSynchronized ... 5231
Volume ... 5232
VolumeHigh ... 5233
VolumeLow ... 5234
Time ... 5235
Spread ... 5236
SpreadFloat ... 5237

© 2000-2024, MetaQuotes Ltd.

TicksBookDepth 5238
StopsLevel 5239
FreezeLevel 5240
Bid 5241
BidHigh 5242
BidLow 5243
Ask 5244
AskHigh 5245
AskLow 5246
Last 5247
LastHigh 5248
LastLow 5249
TradeCalcMode 5250
TradeCalcModeDescription 5251
TradeMode 5252
TradeModeDescription 5253
TradeExecution 5254
TradeExecutionDescription 5255
SwapMode 5256
SwapModeDescription 5257
SwapRollover3days 5258
SwapRollover3daysDescription 5259
MarginInitial 5260
MarginMaintenance 5261
MarginLong 5262
MarginShort 5263
MarginLimit 5264
MarginStop 5265
MarginStopLimit 5266
TradeTimeFlags 5267
TradeFillFlags 5268
Digits 5269
Point 5270
TickValue 5271
TickValueProfit 5272
TickValueLoss 5273
TickSize 5274
ContractSize 5275
LotsMin 5276
LotsMax 5277
LotsStep 5278
LotsLimit 5279
SwapLong 5280
SwapShort 5281
CurrencyBase 5282
CurrencyProfit 5283
CurrencyMargin 5284
Bank 5285
Description 5286
Path 5287
SessionDeals 5288
SessionBuyOrders 5289
SessionSellOrders 5290
SessionTurnover 5291
SessionInterest 5292
SessionBuyOrdersVolume 5293
SessionSellOrdersVolume 5294
SessionOpen 5295

© 2000-2024, MetaQuotes Ltd.

SessionClose ... 5296
SessionAW ... 5297
SessionPriceSettlement ... 5298
SessionPriceLimitMin ... 5299
SessionPriceLimitMax ... 5300
InfoInteger ... 5301
InfoDouble ... 5302
InfoString ... 5303
NormalizePrice ... 5304
COrderInfo ... 5305
Ticket ... 5307
TimeSetup ... 5308
TimeSetupMsc ... 5309
OrderType ... 5310
TypeDescription ... 5311
State ... 5312
StateDescription ... 5313
TimeExpiration ... 5314
TimeDone ... 5315
TimeDoneMsc ... 5316
TypeFilling ... 5317
TypeFillingDescription ... 5318
TypeTime ... 5319
TypeTimeDescription ... 5320
Magic ... 5321
PositionId ... 5322
VolumeInitial ... 5323
VolumeCurrent ... 5324
PriceOpen ... 5325
StopLoss ... 5326
TakeProfit ... 5327
PriceCurrent ... 5328
PriceStopLimit ... 5329
Symbol ... 5330
Comment ... 5331
InfoInteger ... 5332
InfoDouble ... 5333
InfoString ... 5334
StoreState ... 5335
CheckState ... 5336
Select ... 5337
SelectByIndex ... 5338
CHistoryOrderInfo ... 5339
TimeSetup ... 5341
TimeSetupMsc ... 5342
OrderType ... 5343
TypeDescription ... 5344
State ... 5345
StateDescription ... 5346
TimeExpiration ... 5347
TimeDone ... 5348
TimeDoneMsc ... 5349
TypeFilling ... 5350
TypeFillingDescription ... 5351
TypeTime ... 5352
TypeTimeDescription ... 5353
Magic ... 5354
PositionId ... 5355

© 2000-2024, MetaQuotes Ltd.

VolumeInitial ... 5356
VolumeCurrent ... 5357
PriceOpen ... 5358
StopLoss ... 5359
TakeProfit ... 5360
PriceCurrent ... 5361
PriceStopLimit ... 5362
Symbol ... 5363
Comment ... 5364
InfoInteger ... 5365
InfoDouble ... 5366
InfoString ... 5367
Ticket ... 5368
SelectByIndex ... 5369
CPositionInfo ... 5370
Time ... 5372
TimeMsc ... 5373
TimeUpdate ... 5374
TimeUpdateMsc ... 5375
PositionType ... 5376
TypeDescription ... 5377
Magic ... 5378
Identifier ... 5379
Volume ... 5380
PriceOpen ... 5381
StopLoss ... 5382
TakeProfit ... 5383
PriceCurrent ... 5384
Commission ... 5385
Swap ... 5386
Profit ... 5387
Symbol ... 5388
Comment ... 5389
InfoInteger ... 5390
InfoDouble ... 5391
InfoString ... 5392
Select ... 5393
SelectByIndex ... 5394
SelectByMagic ... 5395
SelectByTicket ... 5396
StoreState ... 5397
CheckState ... 5398
CDealInfo ... 5399
Order ... 5401
Time ... 5402
TimeMsc ... 5403
DealType ... 5404
TypeDescription ... 5405
Entry ... 5406
EntryDescription ... 5407
Magic ... 5408
PositionId ... 5409
Volume ... 5410
Price ... 5411
Commision ... 5412
Swap ... 5413
Profit ... 5414
Symbol ... 5415

© 2000-2024, MetaQuotes Ltd.

Comment ... 5416
InfoInteger ... 5417
InfoDouble ... 5418
InfoString ... 5419
Ticket ... 5420
SelectByIndex ... 5421
CTrade ... 5422
LogLevel ... 5426
SetExpertMagicNumber ... 5427
SetDeviationInPoints ... 5428
SetTypeFilling ... 5429
SetTypeFillingBySymbol ... 5430
SetAsyncMode ... 5431
SetMarginMode ... 5432
OrderOpen ... 5433
OrderModify ... 5435
OrderDelete ... 5436
PositionOpen ... 5437
PositionModify ... 5438
PositionClose ... 5439
PositionClosePartial ... 5440
PositionCloseBy ... 5442
Buy ... 5443
Sell ... 5444
BuyLimit ... 5445
BuyStop ... 5447
SellLimit ... 5448
SellStop ... 5449
Request ... 5450
RequestAction ... 5451
RequestActionDescription ... 5452
RequestMagic ... 5453
RequestOrder ... 5454
RequestSymbol ... 5455
RequestVolume ... 5456
RequestPrice ... 5457
RequestStopLimit ... 5458
RequestSL ... 5459
RequestTP ... 5460
RequestDeviation ... 5461
RequestType ... 5462
RequestTypeDescription ... 5463
RequestTypeFilling ... 5464
RequestTypeFillingDescription ... 5465
RequestTypeTime ... 5466
RequestTypeTimeDescription ... 5467
RequestExpiration ... 5468
RequestComment ... 5469
RequestPosition ... 5470
RequestPositionBy ... 5471
Result ... 5472
ResultRetcode ... 5473
ResultRetcodeDescription ... 5474
ResultDeal ... 5475
ResultOrder ... 5476
ResultVolume ... 5477
ResultPrice ... 5478
ResultBid ... 5479

© 2000-2024, MetaQuotes Ltd.

ResultAsk ... 5480
ResultComment ... 5481
CheckResult ... 5482
CheckResultRetcode ... 5483
CheckResultRetcodeDescription ... 5484
CheckResultBalance ... 5485
CheckResultEquity ... 5486
CheckResultProfit ... 5487
CheckResultMargin ... 5488
CheckResultMarginFree ... 5489
CheckResultMarginLevel ... 5490
CheckResultComment ... 5491
PrintRequest ... 5492
PrintResult ... 5493
FormatRequest ... 5494
FormatRequestResult ... 5495
CTerminalInfo ... 5496
Build ... 5498
IsConnected ... 5499
IsDLLsAllowed ... 5500
IsTradeAllowed ... 5501
IsEmailEnabled ... 5502
IsFtpEnabled ... 5503
MaxBars ... 5504
CodePage ... 5505
CPUCores ... 5506
MemoryPhysical ... 5507
MemoryTotal ... 5508
MemoryAvailable ... 5509
MemoryUsed ... 5510
IsX64 ... 5511
OpenCLSupport ... 5512
DiskSpace ... 5513
Language ... 5514
Name ... 5515
Company ... 5516
Path ... 5517
DataPath ... 5518
CommonDataPath ... 5519
InfoInteger ... 5520
InfoString ... 5521
Módulos de estratégias ... 5522
Classes Base para Expert Advisors ... 5525
CExpertBase ... 5526
InitPhase ... 5528
TrendType ... 5529
UsedSeries ... 5530
EveryTick ... 5531
Open ... 5532
High ... 5533
Low ... 5534
Close ... 5535
Spread ... 5536
Time ... 5537
TickVolume ... 5538
RealVolume ... 5539
Init ... 5540
Symbol ... 5541

© 2000-2024, MetaQuotes Ltd.

Period ... 5542
Magic ... 5543
ValidationSettings ... 5544
SetPriceSeries ... 5545
SetOtherSeries ... 5546
InitIndicators ... 5547
InitOpen ... 5548
InitHigh ... 5549
InitLow ... 5550
InitClose ... 5551
InitSpread ... 5552
InitTime ... 5553
InitTickVolume ... 5554
InitRealVolume ... 5555
PriceLevelUnit ... 5556
StartIndex ... 5557
CompareMagic ... 5558
CExpert ... 5559
Init ... 5564
Magic ... 5565
InitSignal ... 5566
InitTrailing ... 5567
InitMoney ... 5568
InitTrade ... 5569
Deinit ... 5570
OnTickProcess ... 5571
OnTradeProcess ... 5572
OnTimerProcess ... 5573
OnChartEventProcess ... 5574
OnBookEventProcess ... 5575
MaxOrders ... 5576
Signal ... 5577
ValidationSettings ... 5578
InitIndicators ... 5579
OnTick ... 5580
OnTrade ... 5581
OnTimer ... 5582
OnChartEvent ... 5583
OnBookEvent ... 5584
InitParameters ... 5585
DeinitTrade ... 5586
DeinitSignal ... 5587
DeinitTrailing ... 5588
DeinitMoney ... 5589
DeinitIndicators ... 5590
Refresh ... 5591
Processing ... 5592
SelectPosition ... 5594
CheckOpen ... 5595
CheckOpenLong ... 5596
CheckOpenShort ... 5597
OpenLong ... 5598
OpenShort ... 5599
CheckReverse ... 5600
CheckReverseLong ... 5601
CheckReverseShort ... 5602
ReverseLong ... 5603
ReverseShort ... 5604

© 2000-2024, MetaQuotes Ltd.

CheckClose ... 5605
CheckCloseLong ... 5606
CheckCloseShort ... 5607
CloseAll ... 5608
Close ... 5609
CloseLong ... 5610
CloseShort ... 5611
CheckTrailingStop ... 5612
CheckTrailingStopLong ... 5613
CheckTrailingStopShort ... 5614
TrailingStopLong ... 5615
TrailingStopShort ... 5616
CheckTrailingOrderLong ... 5617
CheckTrailingOrderShort ... 5618
TrailingOrderLong ... 5619
TrailingOrderShort ... 5620
CheckDeleteOrderLong ... 5621
CheckDeleteOrderShort ... 5622
DeleteOrders ... 5623
DeleteOrder ... 5624
DeleteOrderLong ... 5625
DeleteOrderShort ... 5626
LotOpenLong ... 5627
LotOpenShort ... 5628
LotReverse ... 5629
PrepareHistoryDate ... 5630
HistoryPoint ... 5631
CheckTradeState ... 5632
WaitEvent ... 5633
NoWaitEvent ... 5634
TradeEventPositionStopTake ... 5635
TradeEventOrderTriggered ... 5636
TradeEventPositionOpened ... 5637
TradeEventPositionVolumeChanged ... 5638
TradeEventPositionModified ... 5639
TradeEventPositionClosed ... 5640
TradeEventOrderPlaced ... 5641
TradeEventOrderModified ... 5642
TradeEventOrderDeleted ... 5643
TradeEventNotIdentified ... 5644
TimeframeAdd ... 5645
TimeframesFlags ... 5646
CExpertSignal ... 5647
BasePrice ... 5650
UsedSeries ... 5651
Weight ... 5652
PatternsUsage ... 5653
General ... 5654
Ignore ... 5655
Invert ... 5656
ThresholdOpen ... 5657
ThresholdClose ... 5658
PriceLevel ... 5659
StopLevel ... 5660
TakeLevel ... 5661
Expiration ... 5662
Magic ... 5663
ValidationSettings ... 5664

© 2000-2024, MetaQuotes Ltd.

InitIndicators ... 5665
AddFilter ... 5666
CheckOpenLong ... 5667
CheckOpenShort ... 5668
OpenLongParams ... 5669
OpenShortParams ... 5670
CheckCloseLong ... 5671
CheckCloseShort ... 5672
CloseLongParams ... 5673
CloseShortParams ... 5674
CheckReverseLong ... 5675
CheckReverseShort ... 5676
CheckTrailingOrderLong ... 5677
CheckTrailingOrderShort ... 5678
LongCondition ... 5679
ShortCondition ... 5680
Direction ... 5681
CExpertTrailing ... 5682
CheckTrailingStopLong ... 5684
CheckTrailingStopShort ... 5685
CExpertMoney ... 5686
Percent ... 5688
ValidationSettings ... 5689
CheckOpenLong ... 5690
CheckOpenShort ... 5691
CheckReverse ... 5692
CheckClose ... 5693
Módulos de Sinais de Negociação ... 5694
Signals of the Indicator Accelerator Oscillator ... 5697
Signals of the Indicator Adaptive Moving Average ... 5700
Signals of the Indicator Awesome Oscillator ... 5704
Signals of the Oscillator Bears Power ... 5708
Signals of the Oscillator Bulls Power ... 5710
Signals of the Oscillator Commodity Channel Index ... 5712
Signals of the Oscillator DeMarker ... 5716
Signals of the Indicator Double Exponential Moving Average ... 5720
Signals of the Indicator Envelopes ... 5724
Signals of the Indicator Fractal Adaptive Moving Average ... 5727
Signals of the Intraday Time Filter ... 5731
Signals of the Oscillator MACD ... 5733
Signals of the Indicator Moving Average ... 5739
Signals of the Indicator Parabolic SAR ... 5743
Signals of the Oscillator Relative Strength Index ... 5745
Signals of the Oscillator Relative Vigor Index ... 5751
Signals of the Oscillator Stochastic ... 5753
Signals of the Oscillator Triple Exponential Average ... 5758
Signals of the Indicator Triple Exponential Moving Average ... 5762
Signals of the Oscillator Williams Percent Range ... 5766
Classes de Trailing Stop ... 5769
CTrailingFixedPips ... 5770
StopLevel ... 5772
ProfitLevel ... 5773
ValidationSettings ... 5774
CheckTrailingStopLong ... 5775
CheckTrailingStopShort ... 5776
CTrailingMA ... 5777
Period ... 5779
Shift ... 5780

© 2000-2024, MetaQuotes Ltd.

Method ... 5781
Applied ... 5782
InitIndicators ... 5783
ValidationSettings ... 5784
CheckTrailingStopLong ... 5785
CheckTrailingStopShort ... 5786
CTrailingNone ... 5787
CheckTrailingStopLong ... 5788
CheckTrailingStopShort ... 5789
CTrailingPSAR ... 5790
Step ... 5792
Maximum ... 5793
InitIndicators ... 5794
CheckTrailingStopLong ... 5795
CheckTrailingStopShort ... 5796
Classes de Gerenciamento de Dinheiro ... 5797
CMoneyFixedLot ... 5798
Lots ... 5800
ValidationSettings ... 5801
CheckOpenLong ... 5802
CheckOpenShort ... 5803
CMoneyFixedMargin ... 5804
CheckOpenLong ... 5805
CheckOpenShort ... 5806
CMoneyFixedRisk ... 5807
CheckOpenLong ... 5808
CheckOpenShort ... 5809
CMoneyNone ... 5810
ValidationSettings ... 5811
CheckOpenLong ... 5812
CheckOpenShort ... 5813
CMoneySizeOptimized ... 5814
DecreaseFactor ... 5816
ValidationSettings ... 5817
CheckOpenLong ... 5818
CheckOpenShort ... 5819
Painéis e caixas de diálogo ... 5820
CRect ... 5822
Left ... 5823
Top ... 5824
Right ... 5825
Bottom ... 5826
Width ... 5827
Height ... 5828
SetBound ... 5829
Move ... 5830
Shift ... 5831
Contains ... 5832
Format ... 5833
CDateTime ... 5834
MonthName ... 5836
ShortMonthName ... 5837
DayName ... 5838
ShortDayName ... 5839
DaysInMonth ... 5840
DateTime ... 5841
Date ... 5842
Time ... 5843

© 2000-2024, MetaQuotes Ltd.

Sec ... 5844
Min ... 5845
Hour ... 5846
Day ... 5847
Mon ... 5848
Year ... 5849
SecDec ... 5850
SecInc ... 5851
MinDec ... 5852
MinInc ... 5853
HourDec ... 5854
HourInc ... 5855
DayDec ... 5856
DayInc ... 5857
MonDec ... 5858
MonInc ... 5859
YearDec ... 5860
YearInc ... 5861
CWnd ... 5862
Create ... 5866
Destroy ... 5867
OnEvent ... 5868
OnMouseEvent ... 5869
Name ... 5870
ControlsTotal ... 5871
Control ... 5872
ControlFind ... 5873
Rect ... 5874
Left ... 5875
Top ... 5876
Right ... 5877
Bottom ... 5878
Width ... 5879
Height ... 5880
Move ... 5881
Shift ... 5882
Resize ... 5883
Contains ... 5884
Alignment ... 5885
Align ... 5886
Id ... 5887
IsEnabled ... 5888
Enable ... 5889
Disable ... 5890
IsVisible ... 5891
Visible ... 5892
Show ... 5893
Hide ... 5894
IsActive ... 5895
Activate ... 5896
Deactivate ... 5897
StateFlags ... 5898
StateFlagsSet ... 5899
StateFlagsReset ... 5900
PropFlags ... 5901
PropFlagsSet ... 5902
PropFlagsReset ... 5903
MouseX ... 5904

© 2000-2024, MetaQuotes Ltd.

MouseY ... 5905
MouseFlags ... 5906
MouseFocusKill ... 5907
OnCreate ... 5908
OnDestroy ... 5909
OnMove ... 5910
OnResize ... 5911
OnEnable ... 5912
OnDisable ... 5913
OnShow ... 5914
OnHide ... 5915
OnActivate ... 5916
OnDeactivate ... 5917
OnClick ... 5918
OnChange ... 5919
OnMouseDown ... 5920
OnMouseUp ... 5921
OnDragStart ... 5922
OnDragProcess ... 5923
OnDragEnd ... 5924
DragObjectCreate ... 5925
DragObjectDestroy ... 5926
CWndObj ... 5927
OnEvent ... 5929
Text ... 5930
Color ... 5931
ColorBackground ... 5932
ColorBorder ... 5933
Font ... 5934
FontSize ... 5935
ZOrder ... 5936
OnObjectCreate ... 5937
OnObjectChange ... 5938
OnObjectDelete ... 5939
OnObjectDrag ... 5940
OnSetText ... 5941
OnSetColor ... 5942
OnSetColorBackground ... 5943
OnSetFont ... 5944
OnSetFontSize ... 5945
OnSetZOrder ... 5946
OnDestroy ... 5947
OnChange ... 5948
CWndContainer ... 5949
Destroy ... 5951
OnEvent ... 5952
OnMouseEvent ... 5953
ControlsTotal ... 5954
Control ... 5955
ControlFind ... 5956
Add ... 5957
Delete ... 5958
Move ... 5959
Shift ... 5960
Id ... 5961
Enable ... 5962
Disable ... 5963
Show ... 5964

© 2000-2024, MetaQuotes Ltd.

Hide ... 5965
MouseFocusKill ... 5966
Save ... 5967
Load ... 5968
OnResize ... 5969
OnActivate ... 5970
OnDeactivate ... 5971
CLabel ... 5972
Create ... 5977
OnSetText ... 5978
OnSetColor ... 5979
OnSetFont ... 5980
OnSetFontSize ... 5981
OnCreate ... 5982
OnShow ... 5983
OnHide ... 5984
OnMove ... 5985
CBmpButton ... 5986
Create ... 5993
Border ... 5994
BmpNames ... 5995
BmpOffName ... 5996
BmpOnName ... 5997
BmpPassiveName ... 5998
BmpActiveName ... 5999
Pressed ... 6000
Locking ... 6001
OnSetZOrder ... 6002
OnCreate ... 6003
OnShow ... 6004
OnHide ... 6005
OnMove ... 6006
OnChange ... 6007
OnActivate ... 6008
OnDeactivate ... 6009
OnMouseDown ... 6010
OnMouseUp ... 6011
CButton ... 6012
Create ... 6019
Pressed ... 6020
Locking ... 6021
OnSetText ... 6022
OnSetColor ... 6023
OnSetColorBackground ... 6024
OnSetColorBorder ... 6025
OnSetFont ... 6026
OnSetFontSize ... 6027
OnCreate ... 6028
OnShow ... 6029
OnHide ... 6030
OnMove ... 6031
OnResize ... 6032
OnMouseDown ... 6033
OnMouseUp ... 6034
CEdit ... 6035
Create ... 6040
ReadOnly ... 6041
TextAlign ... 6042

© 2000-2024, MetaQuotes Ltd.

OnObjectEndEdit ... 6043
OnSetText ... 6044
OnSetColor ... 6045
OnSetColorBackground ... 6046
OnSetColorBorder ... 6047
OnSetFont ... 6048
OnSetFontSize ... 6049
OnSetZOrder ... 6050
OnCreate ... 6051
OnShow ... 6052
OnHide ... 6053
OnMove ... 6054
OnResize ... 6055
OnChange ... 6056
OnClick ... 6057
CPanel ... 6058
Create ... 6063
BorderType ... 6064
OnSetText ... 6065
OnSetColorBackground ... 6066
OnSetColorBorder ... 6067
OnCreate ... 6068
OnShow ... 6069
OnHide ... 6070
OnMove ... 6071
OnResize ... 6072
OnChange ... 6073
CPicture ... 6074
Create ... 6079
Border ... 6080
BmpName ... 6081
OnCreate ... 6082
OnShow ... 6083
OnHide ... 6084
OnMove ... 6085
OnChange ... 6086
CScroll ... 6087
Create ... 6089
OnEvent ... 6090
MinPos ... 6091
MaxPos ... 6092
CurrPos ... 6093
CreateBack ... 6094
CreateInc ... 6095
CreateDec ... 6096
CreateThumb ... 6097
OnClickInc ... 6098
OnClickDec ... 6099
OnShow ... 6100
OnHide ... 6101
OnChangePos ... 6102
OnThumbDragStart ... 6103
OnThumbDragProcess ... 6104
OnThumbDragEnd ... 6105
CalcPos ... 6106
CScrollV ... 6107
CreateInc ... 6113
CreateDec ... 6114

© 2000-2024, MetaQuotes Ltd.

CreateThumb ... 6115
OnResize ... 6116
OnChangePos ... 6117
OnThumbDragStart ... 6118
OnThumbDragProcess ... 6119
OnThumbDragEnd ... 6120
CalcPos ... 6121
CScrollH ... 6122
CreateInc ... 6128
CreateDec ... 6129
CreateThumb ... 6130
OnResize ... 6131
OnChangePos ... 6132
OnThumbDragStart ... 6133
OnThumbDragProcess ... 6134
OnThumbDragEnd ... 6135
CalcPos ... 6136
CWndClient ... 6137
Create ... 6140
OnEvent ... 6141
ColorBackground ... 6142
ColorBorder ... 6143
BorderType ... 6144
VScrolled ... 6145
HScrolled ... 6146
CreateBack ... 6147
CreateScrollV ... 6148
CreateScrollH ... 6149
OnResize ... 6150
OnVScrollShow ... 6151
OnVScrollHide ... 6152
OnHScrollShow ... 6153
OnHScrollHide ... 6154
OnScrollLineDown ... 6155
OnScrollLineUp ... 6156
OnScrollLineLeft ... 6157
OnScrollLineRight ... 6158
Rebound ... 6159
CListView ... 6160
Create ... 6166
OnEvent ... 6167
TotalView ... 6168
AddItem ... 6169
Select ... 6170
SelectByText ... 6171
SelectByValue ... 6172
Value ... 6173
CreateRow ... 6174
OnResize ... 6175
OnVScrollShow ... 6176
OnVScrollHide ... 6177
OnScrollLineDown ... 6178
OnScrollLineUp ... 6179
OnItemClick ... 6180
Redraw ... 6181
RowState ... 6182
CheckView ... 6183
CComboBox ... 6184

© 2000-2024, MetaQuotes Ltd.

Create ... 6190
OnEvent ... 6191
AddItem ... 6192
ListViewItems ... 6193
Select ... 6194
SelectByText ... 6195
SelectByValue ... 6196
Value ... 6197
CreateEdit ... 6198
CreateButton ... 6199
CreateList ... 6200
OnClickEdit ... 6201
OnClickButton ... 6202
OnChangeList ... 6203
ListShow ... 6204
ListHide ... 6205
CCheckBox ... 6206
Create ... 6212
OnEvent ... 6213
Text ... 6214
Color ... 6215
Checked ... 6216
Value ... 6217
CreateButton ... 6218
CreateLabel ... 6219
OnClickButton ... 6220
OnClickLabel ... 6221
CCheckGroup ... 6222
Create ... 6228
OnEvent ... 6229
AddItem ... 6230
Value ... 6231
CreateButton ... 6232
OnVScrollShow ... 6233
OnVScrollHide ... 6234
OnScrollLineDown ... 6235
OnScrollLineUp ... 6236
OnChangeItem ... 6237
Redraw ... 6238
RowState ... 6239
CRadioButton ... 6240
Create ... 6242
OnEvent ... 6243
Text ... 6244
Color ... 6245
State ... 6246
CreateButton ... 6247
CreateLabel ... 6248
OnClickButton ... 6249
OnClickLabel ... 6250
CRadioGroup ... 6251
Create ... 6257
OnEvent ... 6258
AddItem ... 6259
Value ... 6260
CreateButton ... 6261
OnVScrollShow ... 6262
OnVScrollHide ... 6263

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown ... 6264
OnScrollLineUp ... 6265
OnChangeItem ... 6266
Redraw ... 6267
RowState ... 6268
Select ... 6269
CSpinEdit ... 6270
Create ... 6276
OnEvent ... 6277
MinValue ... 6278
MaxValue ... 6279
Value ... 6280
CreateEdit ... 6281
CreateInc ... 6282
CreateDec ... 6283
OnClickInc ... 6284
OnClickDec ... 6285
OnChangeValue ... 6286
CDialog ... 6287
Create ... 6289
OnEvent ... 6290
Caption ... 6291
Add ... 6292
CreateWhiteBorder ... 6293
CreateBackground ... 6294
CreateCaption ... 6295
CreateButtonClose ... 6296
CreateClientArea ... 6297
OnClickCaption ... 6298
OnClickButtonClose ... 6299
ClientAreaVisible ... 6300
ClientAreaLeft ... 6301
ClientAreaTop ... 6302
ClientAreaRight ... 6303
ClientAreaBottom ... 6304
ClientAreaWidth ... 6305
ClientAreaHeight ... 6306
OnDialogDragStart ... 6307
OnDialogDragProcess ... 6308
OnDialogDragEnd ... 6309
CAppDialog ... 6310
Create ... 6313
Destroy ... 6314
OnEvent ... 6315
Run ... 6316
ChartEvent ... 6317
Minimized ... 6318
IniFileSave ... 6319
IniFileLoad ... 6320
IniFileName ... 6321
IniFileExt ... 6322
CreateCommon ... 6323
CreateExpert ... 6324
CreateIndicator ... 6325
CreateButtonMinMax ... 6326
OnClickButtonClose ... 6327
OnClickButtonMinMax ... 6328
OnAnotherApplicationClose ... 6329

© 2000-2024, MetaQuotes Ltd.

Rebound 6330
Minimize 6331
Maximize 6332
CreateInstanceId 6333
ProgramName 6334
SubwinOff 6335
36 Migrando do MQL4 6336
37 Lista de Funções MQL5 6340
38 Lista de Constantes MQL5 6375

© 2000-2024, MetaQuotes Ltd.

Manual MQL5

A MetaQuotes Language 5 (MQL5) é uma linguagem de programação para indicadores técnicos, para robôs de negociação e para aplicativos auxiliares, providenciando automatização da negociação nos mercados financeiros. A MQL5 é uma linguagem moderna de alto nível desenvolvida pela MetaQuotes para a sua própria plataforma de negociação. A sintaxe da linguagem é tão próxima quanto possível da C++ e permite escrever programas no estilo da programação orientada a objetos (POO).

Para escrever programas em MQL5, a plataforma de negociação vem com o ambiente de desenvolvimento do MetaEditor que tem todo um conjunto de modernas ferramentas para escrita de códigos, incluindo modelos, trechos de código, depuração, criação de perfis, preenchimento automático e armazenamento de versão integrado MQL5 Storage.

O suporte e o desenvolvimento da linguagem são realizados no site da MQL5.community. Nele existe uma extensa biblioteca de códigos gratuitos e muitos artigos. Estes artigos abrangem todos os tópicos da negociação moderna: redes neurais, estatísticas e análise, negociação de alta frequência, arbitragem, testes e otimização de estratégias de negociação, uso de robôs para automatizar a negociação e muito mais.

Traders e desenvolvedores de programas MQL5 podem interagir no fórum, realizar encomendas no serviço Freelance, comprar e vender programas seguros no Mercado - uma loja de aplicativos prontos para negociação automática.

A linguagem MQL5 contém funções de negociação especializadas e manipuladores de ventos pré-definidos para escrita de Expert Advisors. Os Expert Advisors gerenciam automaticamente os processos de negociação com base nas regras comerciais estabelecidas neles. Você também pode criar em MQL5 seus próprios indicadores técnicos, scripts e bibliotecas de funções.

O manual MQL5 contém - divididas por categorias - funções, operações, palavras reservadas e outras construções da linguagem. Além disso, permite encontrar a descrição de cada elemento da linguagem. Também o manual descreve as classes da Biblioteca padrão para criar estratégias de negociação, painéis de controle, gráficos personalizados e trabalhar com arquivos.

Separadamente do manual, o CodeBase dispõe de uma biblioteca de análise numérica ALGLIB que permite resolver inúmeros problemas matemáticos.

Tipos de aplicativos em MQL5

A fim de executar tarefas específicas para automatizar operações de negociação, os programas MQL5 são divididos em cinco tipos especializados:

- **Expert Advisor** — sistema de negociação automática que está ligado a um gráfico específico. O Expert Advisor contém as funções manipuladoras dos eventos predefinidos que, ao ocorrerem, desencadeiam os respectivos elementos da estratégia de negociação. Esses eventos podem ser a inicialização e a desinicialização do programa, a chegada de um novo tick, a ativação do temporizador, a mudança no livro de ofertas, nos eventos do gráfico e nos eventos do usuário.
  O Expert Advisor pode, além de calcular sinais de negociação segundo as regras estabelecidas, realizar trades automaticamente na conta de negociação, direcionando-os diretamente para o servidor de negociação. Os Expert Advisor são armazenados no *<diretório_do_terminal>\MQL5\Experts*.

© 2000-2024, MetaQuotes Ltd.

- **Indicador personalizado** — indicador técnico escrito pelo usuário, além dos indicadores já integrados na plataforma de negociação. Os indicadores personalizados, assim como os indicadores embutidos, não podem ser trocados automaticamente e se destinam apenas à implementação de funções analíticas. Os indicadores personalizados podem usar os valores de outros indicadores em seus cálculos, e eles próprios podem ser chamados nos próprios Expert Advisors.
  Os indicadores personalizados são armazenados no *<diretório_do_terminal>\MQL5\Indicators*.
- **Script** — programa projetado para a execução única de certa ação. Ao contrário dos Expert Advisors, os scripts não lidam com eventos, exceto com aqueles que são de inicialização e desinicialização. Para executar o script em seu código, deve haver uma função manipulador OnStart. Os scripts são armazenados no *<diretório_do_terminal>\MQL5\Scripts*.
- **Serviço** – um programa que, ao contrário de indicadores, EAs e scripts, não estão vinculados a um gráfico para o seu trabalho. Como os scripts, os serviços processam apenas eventos de inicialização. Para executar o serviço em seu código, deve estar presenta uma função manipulador OnStart. Os serviços aceitam apenas o evento o Start, mas podem enviar aos gráficos eventos personalizados usando EventChartCustom. Os serviços são armazenados no diretório <diretório_do_terminal>\MQL5\Services.
- **Biblioteca** — biblioteca de funções personalizadas projetada para armazenar e distribuir blocos - usados com frequência - de programas do usuário. As bibliotecas não podem ser executadas por conta própria.
  As bibliotecas são armazenadas no *<As bibliotecas não podem ser executadas por conta própria.>\MQL5\Libraries*
- **Arquivo de inclusão** — código fonte de blocos - usados com frequência - de programas do usuário. Esses arquivos podem ser incluídos nos códigos fonte de Expert Advisors, de scripts, de indicadores personalizados e de bibliotecas durante fase de compilação. Devido à sobrecarga adicional ao chamar funções de biblioteca, é melhor usar arquivos de inclusão do que bibliotecas.
  Arquivos de inclusão podem estar no mesmo diretório que o código fonte, neste caso é usada a diretiva #include com aspas duplas. Outro local para armazenar os arquivos de inclusão está no *<diretório_do_terminal>\MQL5\Include*, neste caso, é utilizada a diretiva *#include* com colchetes angulares.

© 2000-2024, MetaQuotes Ltd

Elementos Básicos da Linguagem

A Linguagem MetaQuotes 5 (MQL5) é uma linguagem orientado a objetos de alto nível destinado para escrever estratégias de negociação automatizadas, e indicadores técnicos customizados para a a análise de vários mercados financeiros. Ele permite não somente escrever uma variedade de sistemas especialistas, concebidos para operar em tempo real, mas também criar suas próprias ferramentas gráficas para ajudar você a tomar decisões de negociação (trade).

MQL5 é baseado no conceito da popular linguagem de programação C++. Em comparação com MQL4, a nova linguagem tem agora enumeradores, estruturas, classes e manipulação de evento. Ao aumentar o número de tipos principais incorporados, a interação dos programas executáveis em MQL5 com outros aplicativos através de dll é agora tão fácil quanto possível. A sintaxe do MQL5 é similar à sintaxe do C++, e isso faz com que seja fácil traduzir para ele programas de linguagens modernas de programação.

Para ajudar você a estudar a linguagem MQL5, todos os tópicos estão agrupados nas seguintes seções:

- Sintaxe
- Tipos de Dados
- Operações e Expressões
- Operadores
- Funções
- Variáveis
- Preprocessador
- Programação Orientada a Objetos
- Espaços de nomes

© 2000-2024, MetaQuotes Ltd.

Sintaxe

Quanto à sintaxe, a linguagem MQL5 para programação de estratégias de negociação é muito similar à linguagem de programação C++, com exceção de alguns recursos:

- ausência de aritmética com endereço
- ausência do operador goto
- um enumerador anônimo não pode ser declarado
- ausência de herança múltipla

Também Veja

Enumeradores, Estruturas e Classes, Herança

© 2000-2024, MetaQuotes Ltd.

Comentários

Comentários de múltiplas linhas começam com o par de símbolos /* e terminam com o par */. Este tipo de comentário não pode ser aninhado. Comentários de linha única começam com o par de símbolos // e terminam com o caractere newline, eles podem ser aninhados em outros comentários de múltiplas linhas. Comentários são permitidos em todos os lugares onde os espaços são permitidos, eles podem conter qualquer número de espaços.

Exemplos:

```
//--- Comentário de linha única
/*  Comentário
   de múltiplas         // Comentário de linha única aninhada
   linhas
*/
```

© 2000-2024, MetaQuotes Ltd.

Identificadores

Identificadores são usados como nomes de variáveis e funções. O comprimento de um identificador não pode exceder 63 caracteres.

Caracteres permitidos na formação de um identificador: números 0-9, as letras maiúsculas e minúsculas latinas a-z e A-Z, reconhecidos como caracteres diferentes, o caractere sublinhado (_). O primeiro caractere não pode ser um dígito.

O identificador não deve coincidir com uma palavra reservada.

Exemplos:

```
NAME1 name1 Total_5 Paper
```

Também Veja

Variáveis, Funções

© 2000-2024, MetaQuotes Ltd.

Palavras Reservadas

Os seguintes identificadores são registrados como palavras reservadas, cada uma delas corresponde a uma certa ação, e não pode ser usada com outro significado:

Tipos de Dados

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

Especificadores de Acesso

const	private	virtual
delete	protected	
override	public	

Classes de Memória

extern	input	static

Operadores

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

Outros

© 2000-2024, MetaQuotes Ltd.

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

© 2000-2024, MetaQuotes Ltd.

Tipos de Dados

Qualquer programa funciona com dados. Os dados podem ser de diferentes tipos dependendo da sua finalidade. Por exemplo, dados inteiros são usados para acessar componentes de arrays. Dados de preço pertencem ao tipo de dupla precisão com ponto flutuante. Isso está relacionado ao fato de que nenhum tipo de dados especial para detalhes de preço é fornecido pelo MQL5.

Dados de diferentes tipos são processados com diferentes taxas. Dados inteiros são os de processamento mais rápido. Para processar dados de dupla precisão, um co-processador especial é utilizado. No entanto, por causa da complexidade da representação interna de dados com ponto flutuante, eles são processados mais devagar que os dados inteiros.

Dados do tipo string são os de processamento mais lento por causa da alocação/realocação dinâmica de memória de computador.

Os tipos básicos de dado são:

- inteiros (char, short, int, long, uchar, ushort, uint, ulong);
- lógico (bool);
- literais (ushort);
- strings (string);
- números com ponto flutuante (double, float);
- cor (color);
- data e hora(datetime);
- enumeradores (enum).

Os tipos complexos de dado são:

- estruturas;
- classes.

Em termos de OOP tipos de dados complexos são chamados de tipos de dados abstratos.

Os tipos *color* e *datetime* fazem sentido apenas para facilitar a entrada e visualização de parâmetros definidos do lado de fora - a partir da tabela de um Expert Advisor ou propriedades de um indicador customizado (o guia Entradas). Dados de *cor (color)* e *data e hora (datetime)* são representados como inteiros. Tipos inteiros e tipos de ponto flutuante são chamados tipos aritméticos (numéricos).

Apenas conversões de tipo (type casting) implícitas são usadas em expressões, a menos que uma conversão (casting) explícita seja especificado.

Também Veja

Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

Tipos Inteiros

Em MQL5, os inteiros são representados por onze tipos. Alguns tipos podem ser utilizados em conjunto com outros tipos, se exigido pela lógica do programa, mas neste caso é necessário lembrar as regras de conversão de tipos.

A tabela abaixo lista as características de cada tipo. Além disso, a última coluna apresenta o tipo correspondente em C++ para cada tipo.

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

Valores do tipo integer podem ser apresentados como constantes numéricas, literais de cor, literais de data-hora, constantes de caractere e enumeradores.

Também Veja

Conversão de Dados, Constantes de Tipos Numéricos

© 2000-2024, MetaQuotes Ltd.

Tipos char, short, int e long

char

O tipo *char* usa 1 byte de memória (8 bits) e permite expressar em notação binária 2^8=256 valores. O tipo *char* pode conter tanto valores positivos quanto negativos. A faixa de valores é de -128 a 127.

uchar

O tipo inteiro *uchar* também ocupa 1 byte de memória, assim como o tipo *char* , mas diferente dele *uchar* é destinado apenas para valores positivos. O valor mínimo é zero, o valor máximo é 255. A primeira letra u no nome do tipo *uchar* é abreviatura de *unsigned (sem sinal)*.

short

O tamanho do tipo *short* é de 2 bytes (16 bits) e, conseqüentemente, ele permite expressar a faixa de valores igual a 2 elevado a 16: 2^16 = 65 536. Como o tipo *short* é um tipo com sinal, e contém tanto valores positivos quanto negativos, a faixa de valores é entre -32 768 e 32 767.

ushort

O tipo *short* sem sinal é o tipo *ushort*, que também tem 2 bytes de tamanho. O valor mínimo é 0, o valor máximo é 65 535.

int

O tamanho do tipo *int* é de 4 bytes (32 bits). O valor mínimo é -2 147 483 648, o valor máximo é 2 147 483 647.

uint

O tipo integer sem sinal é *uint*. Ele usa 4 bytes de memória e permite expressar inteiros de 0 a 4 294 967 295.

long

O tamanho do tipo *long* é de 8 bytes (64 bits). O valor mínimo é -9 223 372 036 854 775 808, o valor máximo é 9 223 372 036 854 775 807.

ulong

O tipo *ulong* também ocupa 8 bytes e pode armazenar valores de 0 a 18 446 744 073 709 551 615.

Exemplos:

```
char  ch=12;
short sh=-5000;
int   in=2445777;
```

© 2000-2024, MetaQuotes Ltd.

Como os tipo inteiros sem sinal não são concebidos para armazenar valores negativos, a tentativa de atribuir um valor negativo pode levar a conseqüências inesperadas. Este simples script levará a um loop infinito:

```
//--- Loop infinito
void OnStart()
  {
   uchar  u_ch;

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

A variante correta é:

```
//--- Variante correta
void OnStart()
  {
   uchar  u_ch;

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

Resultado:

```
   ch= -128  u_ch= 128
   ch= -127  u_ch= 129
   ch= -126  u_ch= 130
   ch= -125  u_ch= 131
   ch= -124  u_ch= 132
   ch= -123  u_ch= 133
   ch= -122  u_ch= 134
   ch= -121  u_ch= 135
   ch= -120  u_ch= 136
   ch= -119  u_ch= 137
   ch= -118  u_ch= 138
   ch= -117  u_ch= 139
   ch= -116  u_ch= 140
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
```

© 2000-2024, MetaQuotes Ltd.

```
   ch= -111  u_ch= 145
    ...
```

Exemplos:

```
//--- Valores negativos não podem ser armazenados em tipos sem sinal
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Hexadecimal: números 0-9, as letras a-f ou A-F para os valores de 10-15; começam com 0x ou 0X.

Exemplos:

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

For integer variables, the values can be set in binary form using B prefix. For example, you can encode the working hours of a trading session into int type variable and use information about them according to the required algorithm:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- set 1 for working hours and 0 for nonworking ones
   int AsianSession   =B'111111111'; // Asian session from 0:00 to 9:00
   int EuropeanSession=B'11111111000000000'; // European session 9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // American session 16:00 - 02:0
//--- derive numerical values of the sessions
   PrintFormat("Asian session hours as value =%d",AsianSession);
   PrintFormat("European session hours as value is %d",EuropeanSession);
   PrintFormat("American session hours as value is %d",AmericanSession);
//--- and now let's display string representations of the sessions' working hours
   Print("Asian session ",GetHoursForSession(AsianSession));
   Print("European session ",GetHoursForSession(EuropeanSession));
   Print("American session ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| return the session's working hours as a string                   |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- in order to check, use AND bit operations and left shift by 1 bit <<=1
//--- start checking from the lowest bit
   int bit=1;
   string out="working hours: ";
//--- check all 24 bits starting from the zero one and up to 23 inclusively
   for(int i=0;i<24;i++)
```

```
      {
       //--- receive bit state in number
       bool workinghour=(session&bit)==bit;
       //--- add the hour's number to the message
       if(workinghour )out=out+StringFormat("%d ",i);
       //--- shift by one bit to the left to check the value of the next one
       bit<<=1;
      }
//--- result string
   return out;
  }
```

Também Veja

Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

Constantes de Caracteres

Os caracteres como elementos de uma string,em MQL5, são índices do conjunto de caracteres Unicode. Eles são valores hexadecimais que podem ser convertidos para inteiros, e que podem ser manipulados por operações com inteiros como adição e subtração.

Qualquer caractere único entre aspas simples ou um código hexadecimal ASCII de um caractere como '\x10' é um caractere constante e é do tipo ushort. Por exemplo, uma anotação do tipo '0' representa o valor numérico 30, que corresponde ao índice do zero na tabela de caracteres.

Exemplo:

```
void OnStart()
  {
//--- define constantes de caracteres
   int symbol_0='0';
   int symbol_9=symbol_0+9; // obtém o símbolo '9'
//--- valores de saída de constantes
   printf("Num formato decimal: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("Num formato hexadecimal: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbol
//--- entrada de constantes em uma string
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- isso é como eles se apresentam em uma string
   Print(test);
  }
```

Uma barra invertida (\) é um caractere de controle para o compilador ao lidar com strings constantes e constantes caracteres no texto fonte de um programa. Alguns símbolos, por exemplo uma aspas simples ('), aspas duplas ("), barra invertida (\) e caracteres de controle podem ser representados como combinação de símbolos que começam com uma barra invertida (\), de acordo com a tabela abaixo:

Nome do Caractere	Código Mnemônico ou imagem	Anotação em MQL5	Valor Numérico
linha nova (line feed)	LF	'\n'	10
tab horizontal	HT	'\t'	9
retorno de carro	CR	'\r'	13
barra invertida	\	'\\'	92
aspas simples	'	'\''	39
aspas duplas	"	'\"'	34
código hexadecimal	hhhh	'\xhhhh'	1 a 4 caracteres hexadecimais

© 2000-2024, MetaQuotes Ltd.

Nome do Caractere	Código Mnemônico ou imagem	Anotação em MQL5	Valor Numérico
código decimal	d	'\d'	número decimal de 0 a 65535

Caso uma barra invertida seja seguida por outro caractere não descrito acima, o resultado é indefinido.

Exemplo

```
void OnStart()
  {
//--- declara constantes de caracteres
   int a='A';
   int b='$';
   int c='©';       // código 0xA9
   int d='\xAE';    // código do símbolo ®
//--- saída imprime constantes
   Print(a,b,c,d);
//--- acrescenta um caractere na string
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,b);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,c);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,d);
   Print(test);
//--- representa caracteres como número
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- acrescenta um caractere na string
   StringSetCharacter(test,1,a1);
   Print(test);
//--- acrescenta um caractere na string
   StringSetCharacter(test,1,b1);
   Print(test);
//--- acrescenta um caractere na string
   StringSetCharacter(test,1,c1);
   Print(test);
//--- acrescenta um caractere na string
   StringSetCharacter(test,1,d1);
```

© 2000-2024, MetaQuotes Ltd.

```
   Print(test);
  }
```

Como foi mencionado acima, o valor de uma constante caractere (ou variável) é o valor do índice na tabela de caracteres. Sendo o índice um inteiro, pode ser escrito de diferentes maneiras.

```
void OnStart()
  {
//---
   int a=0xAE;      // o código de ® corresponde ao literal '\xAE'
   int b=0x24;      // o código de $ corresponde ao literal '\x24'
   int c=0xA9;      // o código de © corresponde ao literal '\xA9'
   int d=0x263A;    // o código de ☺ corresponde ao literal '\x263A'
//--- mostrar valores
   Print(a,b,c,d);
//--- acrescenta um caractere na string
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,b);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,c);
   Print(test);
//--- substitui um caractere na string
   StringSetCharacter(test,0,d);
   Print(test);
//--- código de terno cartão
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- acrescenta um caractere de espadas
   StringSetCharacter(test,1,a1);
   Print(test);
//--- acrescenta um caractere de copas
   StringSetCharacter(test,2,b1);
   Print(test);
//--- acrescenta um caractere de ouros
   StringSetCharacter(test,3,c1);
   Print(test);
//--- acrescenta um caractere de paus
   StringSetCharacter(test,4,d1);
   Print(test);
//--- Exemplo de literais de caractere em uma string
   test="Rainha\x2660Ás\x2662";
   printf("%s",test);
  }
```

© 2000-2024, MetaQuotes Ltd.

A representação internal de um literal de caractere é do tipo ushort. Constantes de caracteres podem aceitar valores de 0 a 65535.

Também Veja

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

© 2000-2024, MetaQuotes Ltd.

Tipo datetime

O tipo datetime é destinado a armazenar data e hora como o número de segundos decorridos desde 01 de Janeiro de 1970. Este tipo ocupa 8 bytes de memória.

Constantes de data e hora podem ser representados como string literal, que consiste de 6 partes mostrando o valor numérico do ano, mês, dia (ou dia, mês, ano), horas, minutos e segundos. A constante é colocado entre aspas simples e começa com o caractere D.

Os valores variam de 1 de Janeiro de 1970 a 31 de Dezembro de 3000. Tanto a data (ano, mês, dia) quanto a hora (horas, minutos, segundos), ou ambos podem ser omitidos.

Na especificação da data literal , é desejável que você especifique ano, mês e dia. Caso contrário, o compilador retorna um aviso sobre entrada incompleta.

Exemplos:

```
datetime NY=D'2015.01.01 00:00';      // Data Hora de começo do ano 2015
datetime d1=D'1980.07.19 12:30:27';   // Ano Mês Dia Horas Minutos Segundos
datetime d2=D'19.07.1980 12:30:27';   // Igual a D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12';         // Igual a D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004';            // Igual a D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__;              // Data de Compilação
datetime compilation_date_time=__DATETIME__;     // Data e Hora de Compilação
datetime compilation_time=__DATETIME__-__DATE__;// Hora de Compilação
//--- Exemplos de declarações após o qual avisos do compilador serão retornados
datetime warning1=D'12:30:27';        // Igual a D'[data de compilação] 12:30:27'
datetime warning2=D'';                // Igual a __DATETIME__
```

Também Veja

Estrutura do Tipo Data, Data e Hora, TimeToString, StringToTime

© 2000-2024, MetaQuotes Ltd.

Tipo color

O tipo color é destinado para armazenar informações sobre cor e ocupa 4 bytes na memória. O primeiro byte é ignorado, os restantes 3 bytes contém os componentes RGB.

As constantes de cor podem ser representadas de três formas: literalmente, por inteiros, ou por nome (somente por Web-colors nomeados).

A representação literal consiste de três partes representando os valores numéricos das taxas dos três principais componentes de cor: vermelho, verde e azul. A constante começa com C e colocada entre aspas simples. Os valores numéricos da taxa de um componente de cor ficam entre 0 e 255.

A representação dos valores inteiros é escrito na forma de um número hexadecimal ou decimal. Um número hexadecimal tem a forma 0x00BBGGRR, onde RR é a taxa do componente da cor vermelha (Red), GG - da cor verde (Green), e BB - da cor azul (Blue). As constantes decimais não são diretamente refletidos no RGB. Eles representam um valor decimal da representação hexadecimal de inteiros.

Cores específicas refletem o chamado conjunto de Web-colors.

Exemplos:

```
//--- Literais
C'128,128,128'    // Cinza
C'0x00,0x00,0xFF' // Azul
//nomes de cores
clrRed               // Vermelho
clrYellow            // Amarelo
clrBlack             // Preto
//--- Representações integrais
0xFFFFFF          // Branco
16777215          // Branco
0x008000          // Verde
32768             // Verde
```

Também Veja

Web Colors, ColorToString, StringToColor, Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

Tipo bool

O tipo bool é destinado para armazenar os valores lógicos true ou false, a representação numérica deles é 1 ou 0, respectivamente.

Exemplos:

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

A representação internal é um número inteiro de 1 byte de tamanho. Deve-se notar que em expressões lógicas é possível usar outros tipos reais ou inteiros ou expressões desses tipos - o compilador não irá gerar qualquer erro. Neste caso, o valor zero será interpretado como false (falso), e todos os outros valores - como true (verdadeiro).

Exemplos:

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

   if(d) Print("d = ",d," e tem o valor true");
   else Print("d = ",d," e tem o valor false");

   i=0;
   if(i) Print("i = ",i," e tem o valor true");
   else Print("i = ",i,"e tem o valor false");

   d=0.0;
   if(d) Print("d = ",d," e tem o valor true");
   else Print("d = ",d," e tem o valor false");

//--- Resultados da execução
//   i= 5 e tem o valor true
//   d= -2.5 e tem o valor true
//   i= 0 e tem o valor false
//   d= 0 e tem o valor false
```

Também Veja

Operações Booleanas, Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Enumeradores

Dados do tipo enum pertencem a um determinado conjunto limitado de dados. Definindo o tipo enumerador:

```
enum nome do tipo enumerável
  {
  lista de valores
  };
```

A lista de valores é uma lista de identificadores de constantes nomeados separados por vírgulas.

Exemplo:

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

Após a enumeração ser declarada, um novo tipo de dados de inteiro de 4 bytes aparece. A declaração do novo tipo de dados permite ao compilador controlar rigidamente os tipos de parâmetros passados, porque a enumeração introduz novas constantes nomeadas. No exemplo acima, a constante nomeada January tem valor igual a 0, February - 1, December - 11.

Rule: Se um certo valor não estiver atribuído a uma constante nomeada que é membro da enumeração, seu novo valor será formado automaticamente. Se ele é o primeiro membro da enumeração, o valor 0 será atribuído a ele. Para todos os subseqüentes membros, os valores serão calculados baseados no valor dos membros anteriores adicionando um.

Exemplo:

```
enum intervals  // Enumeração de constantes nomeadas
   {
    month=1,      // Intervalo de um mês
    two_months,   // Dois meses
    quarter,      // Três meses - trimestre
    halfyear=6,   // Semestre
    year=12,      // Ano - 12 meses
   };
```

© 2000-2024, MetaQuotes Ltd.

Observações

- Diferente do C++, o tamanho da representação interna do tipo enumerado no MQL5 é sempre igual a 4 bytes. Ou seja, sizeof(months) retorna o valor 4.
- Diferente do C++, uma enumeração anônima não pode ser declarada no MQL5. Ou seja, um nome único deve ser sempre especificado após a palavra-chave enum.

Também Veja

Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

Tipos Reais (double, float)

Tipos Reais (ou tipos de ponto flutuante) representam valores com um parte fracionária. Na linguagem MQL5 existem dois tipos para números de ponto flutuante. O método de representação dos números reais na memória do computador é definido pelo padrão IEEE 754 e é independente de plataformas, sistemas operacionais ou linguagens de programação.

Tipo	Tamanho em bytes	Valor Positivo Mínimo	Valor Máximo	C++ Análogo
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

double

Tipo de número real double ocupa 64 bits (1 bit de sinal, 11 bits de expoente e 52 bits de mantissa).

float

Tipo de número real float ocupa 32 bits (1 bit de sinal, 8 bits de expoente e 23 bits de mantissa).

vector

Array unidimensional de números do tipo double. A memória de dados é alocada dinamicamente. As propriedades vetoriais podem ser obtidas por métodos, e o tamanho do vetor pode ser alterado. Nas funções de modelo é possível usar a notação vector<double>.

vectorf

Array unidimensional de números do tipo float, pode ser usado em vez de vector, se a perda de precisão não for importante. Nas funções de modelo é possível usar a notação vector<float>.

vectorc

Array unidimensional de números do tipo complex, é projetado para trabalhar com números complexos. Nas funções de modelo é possível usar a notação vector<complex>. No momento, as operações sobre vetores do tipo vectorc ainda não foram implementadas.

matrix

Uma matriz é um array bidimensional de números do tipo double. A memória para os elementos da matriz é alocada de forma dinâmica. As propriedades da matriz podem ser obtidas por métodos, e o tamanho do vetor pode ser alterado. Nas funções de modelo é possível usar a notação matrix<double>.

© 2000-2024, MetaQuotes Ltd.

matrixf

Array de números bidimensional do tipo float, pode ser usado no lugar de matrix, se a perda de precisão não for importante. Nas funções de modelo, é possível usar a notação matrix<float>.

matrixc

Array de números bidimensional do tipo complex, é projetado para trabalhar com números complexos. Nas funções de modelo é possível usar a notação matrix<complex>. No momento, as operações sobre matrizes do tipo matrixc ainda não foram implementadas.

O nome double significa que a precisão destes números é duas vezes a precisão dos números do tipo float. Na maioria dos casos, o tipo double é o tipo mais conveniente. Em muitos casos a precisão limitada de números float não é suficiente. O motivo do tipo float ser ainda utilizado é a economia de memória (isto é importante para arrays grandes de números reais).

Constantes de ponto flutuante consistem de um parte inteira, um ponto (.) e a parte fracionária. As partes inteira e fracionária são seqüências de algarismos decimais.

Exemplos:

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

Existe uma forma científica de escrever constantes reais, frequentemente este método de notação é mais compacta que a forma tradicional.

Exemplo:

```
double c1=1.12123515e-25;
double c2=0.000000000000000000000000112123515; // 24 zeros após o ponto decimal

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

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

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

Deve-se lembrar que números reais são armazenados em memória com precisão limitada no sistema binário, apesar da notação decimal ser geralmente usada. É por isso que muitos números que são precisamente representados no sistema decimal só podem ser escritos como fração infinita no sistema binário.

Por exemplo, os números 0.3 e 0.7 são representados no computador como frações infinitas, enquanto o número 0.25 é armazenado de forma exata, porque ele representa uma potência de dois.

© 2000-2024, MetaQuotes Ltd.

Neste sentido, recomenda-se fortemente não comparar dois números com igualdade, porque tal comparação não é correta.

Exemplo:

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

Se você ainda precisa comparar com igualdade dois números reais, então você pode fazer isso de duas maneiras diferentes. A primeira maneira é comparar a diferença entre dois números com alguma quantidade pequena que especifica a precisão da comparação.

Exemplo:

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0)
      epsilon=-epsilon;
//---
   if(d1-d2>epsilon)
      return false;
   if(d1-d2<-epsilon)
      return false;
//---
   return true;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001))
      Print(d_val," equals ",f_val);
   else
      Print("Diferente: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToString
// Resultado: Diferente: d_val= 0.7000000000000000   f_val= 0.6999999880790710
  }
```

Note que o valor de epsilon no exemplo acima pode ser menor que a constante predefinida DBL_EPSILON. O valor desta constante é 2.2204460492503131e-016. A constante correspondente ao

© 2000-2024, MetaQuotes Ltd.

tipo float é FLT_EPSILON = 1.192092896e-07. O significado destes valores é o seguinte: é o menor valor que satisfaz a condição 1.0 + DBL_EPSILON! = 1.0 (para números do tipo float 1.0 + FLT_EPSILON! = 1.0).

A segunda maneira compara a diferença normalizada de dois números reais com zero. Não faz sentido comparar a diferença de números normalizados com zero, porque qualquer operação matemática com números normalizados dá um resultado não normalizado.

Exemplo:

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

Algumas operações do co-processador matemático podem resultar em um número real inválido, o qual não pode ser usado em operações matemáticas e operações de comparação, porque o resultado de operações com números reais inválidos é indefinido. Por exemplo, quando tentar calcular o arco-seno de 2, o resultado é infinito negativo.

Exemplo:

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

Além do menos infinito, existe o mais infinito e o NaN (not a number). Para determinar se um número é inválido, você pode usar MathIsValidNumber(). De acordo com o padrão IEEE, eles tem uma representação de máquina especial. Por exemplo, mais infinito para o tipo double tem a representação binária de 0x7FF0 0000 0000 0000.

Exemplos:

```
struct str1
  {
   double d;
  };
struct str2
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   long l;
  };

//--- Começo
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);        // Obtém o número inválido -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;     // número inválido -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;     // maior sem-número SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;     // menor sem-número QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;     // maior sem-número QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;     // // Infinito positivo 1.#INF e menor sem-número SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;     // Infinito negativo -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;     // Zero negativo -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;     // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;     // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;     // Maior número normalizado (MAX_DBL)
   s1=s2;
```

© 2000-2024, MetaQuotes Ltd.

```
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;     // Menor normalizado positivo (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                    // Mostra que o número 0.7 é uma fração sem fim
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

Também Veja

DoubleToString, NormalizeDouble, Constantes de Tipos Numéricos

© 2000-2024, MetaQuotes Ltd.

Números complexos (complex)

O tipo complex embutido é uma estrutura com dois campos double:

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

O tipo "complex" pode ser passado por valor como um parâmetro para funções MQL5 (ao contrário de estruturas comuns, que são passadas apenas por referência). Para funções importadas a partir de DLLs, o tipo "complex" deve ser passado apenas por referência.

O sufixo 'i' é usado para descrever constantes complexas:

```
complex square(complex c)
  {
   return(c*c);
  }

void OnStart()
  {
   Print(square(1+2i));  // como parâmetro é passada a constante
  }
// será exibido "(-3,4)", que é uma representação de string para um número complexo
```

Para números complexos, atualmente estão disponíveis apenas operações simples: =, +, -, *, /, +=, -=, *=, /=, ==, !=.

No futuro, funções matemáticas adicionais serão adicionadas: obtenção do valor absoluto, seno, cosseno e muitos outros.

vectorc

Array unidimensional de números do tipo complex, é projetado para trabalhar com números complexos. Nas funções de modelo é possível usar a notação vector<complex>. No momento, as operações sobre vetores do tipo vectorc ainda não foram implementadas.

matrix

Uma matriz é um array bidimensional de números do tipo double. A memória para os elementos da matriz é alocada de forma dinâmica. As propriedades da matriz podem ser obtidas por métodos, e o tamanho do vetor pode ser alterado. Nas funções de modelo é possível usar a notação matrix<double>.

matrixf

Array de números bidimensional do tipo float, pode ser usado no lugar de matrix, se a perda de precisão não for importante. Nas funções de modelo, é possível usar a notação matrix<float>.

© 2000-2024, MetaQuotes Ltd.

matrixc

Array de números bidimensional do tipo complex, é projetado para trabalhar com números complexos. Nas funções de modelo é possível usar a notação matrix<complex>. No momento, as operações sobre matrizes do tipo matrixc ainda não foram implementadas.

© 2000-2024, MetaQuotes Ltd.

Tipo string

O tipo string é usado para armazenar cadeias de texto. Uma cadeia de texto é uma seqüência de caracteres no formato Unicode com zero no final do mesmo. Uma constante string pode ser atribuída a uma variável string. Uma constante string é uma seqüência de caracteres entre aspas duplas: "This is a string constant".

Se for preciso incluir um aspas duplo (") em uma string, a barra invertida (\) deve ser colocada antes dele. Quaisquer constantes de caractere especial podem ser escritos em uma string, se a barra invertida (\) for digitada antes deles.

Exemplos:

```
string svar="Esta é uma string";
string svar2=StringSubstr(svar,0,4);
Print("Símbolo de direitos autorais\t\x00A9");
FileWrite(handle,"Esta string contém uma nova linha de caracteres \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Para tornar o código fonte legível, longas strings de constantes podem ser divididas em partes sem a operação de adição. Durante a compilação, estas partes serão combinadas em uma string longa:

```
//--- Declara uma string constante longa
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Trade Operations Report</title>\n"
                    "</head>";
//---Saída da string constante no log
   Print(HTML_head);
  }
```

Métodos incorporados do tipo string

Para manusear strings, você pode usar as funções de strings, as funções de conversão e os métodos intrínsecos do tipo string mostrados na tabela a seguir:

Método string	Análogo	Descrição
Construtor string(const int len)		Constrói uma string do comprimento especificado
string[] tanto para leitura como para escrita. O índice deve estar dentro dos limites de BufferSize()		Fornece acesso ao elemento da string pelo índice especificado

© 2000-2024, MetaQuotes Ltd.

Método string	Análogo	Descrição
static string string.Init(const int len, const ushort character);	StringInit	Inicializa uma string com os caracteres especificados e o tamanho especificado
void string.Fill(const ushort character);	StringFill	Preenche uma string com o caractere especificado
int string.Len();	StringLen	Retorna o número de caracteres na string
int string.BufferSize();	StringBufferLen	Retorna o tamanho do buffer alocado para a string
bool string.SetLen(const int new_len);	StringSetLength	Define o comprimento especificado em caracteres da string
bool string.Reserve(const int buffer_len);	StringReserve	Reserva o buffer do tamanho especificado na memória para a string
bool string.Add(const string substring);	StringAdd	Junta a substring especificada desde o final
int string.Concatenate(const scalar val1, const scalar val2...);	StringConcatenate	Forma uma string a partir dos parâmetros passados
array string.Split(const ushort separator, const bool long_separator);	StringSplit	Retorna uma matriz de strings pelo separador dado
int string.Compare(const string str, const bool case_sensivity);	StringCompare	Compara com a string especificada e retorna 1, caso a primeira string seja maior que a segunda; 0, caso as strings sejam iguais; -1 (menos um), caso a primeira string seja menor que a segunda
string string.Substr(const int start_pos, const int len);	StringSubstr	Extrai uma substring de uma posição especificada
int string.Find(const string substr, const int pos);	StringFind	Retorna o número da posição em que a substring começa
void string.ToLower();	StringToLower	Converte todos os caracteres em minúsculas (pequenas)
void string.ToUpper();	StringToUpper	Converte todos os caracteres em maiúsculas (grandes)
int string.TrimLeft();	StringTrimLeft	Elimina caracteres de retorno de carro, espaços e tabulações à esquerda

© 2000-2024, MetaQuotes Ltd.

Método string	Análogo	Descrição
int string.TrimRight()	StringTrimRight	Elimina caracteres de retorno de carro, espaços e tabulações à direita
void string.Double(const double var, const int digits=8);	DoubleToString	Converte uma string para um número do tipo double
void string.Enum(const enum value);	EnumToString	Converter um valor de enumeração de qualquer tipo em uma string
void string.Integer(const int value, const int str_len=0, const ushort fill=' ');	IntegerToString	Converte uma string para um número do tipo long
void string.CharArray(const uchar array[], const int start_pos=0, const int len=-1, const uint cp=CP_ACP);	CharArrayToString	Converte parte de uma matriz do tipo uchar в строку
void string.ShortArray(const ushort array[], const int start_pos=0, const int len=-1);	ShortArrayToString	Copia parte de uma matriz do tipo ushort в строку
void string.Time(const datetime dt,const int mode=TIME_DATE|TIME_MINUTES);	TimeToString	Conversão do valor datetime em uma string com formato "yyyy.mm.dd hh:mi".
void string.Format(const string format_str);	StringFormat	Formata os parâmetros recebidos em uma string

Veja também

Funções de Conversão, Funções de String, FileOpen, FileReadString, FileWriteString

© 2000-2024, MetaQuotes Ltd.

Estruturas, Classes e Interfaces

Estruturas

Uma estrutura é um conjunto de elementos de qualquer tipo (exceto o tipo void). Portanto, a estrutura combina dados logicamente relacionados de diferentes tipos.

Declaração da Estrutura

O tipo de dados estrutura é determinado pela seguinte descrição:

```
struct structure_name
  {
   elements_description
  };
```

O nome da estrutura não pode ser usado como um identificador (nome de uma variável ou função). Deve-se notar que em estruturas MQL5, os elementos seguem um ao outro diretamente, sem alinhamento. Em C++ tal ordem é feita pelo compilador usando a seguinte instrução:

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

Se você quiser ter outro alinhamento na estrutura, use membros auxiliares, "fillers" (preenchedores) para o tamanho certo.

Exemplo:

```
struct trade_settings
  {
   uchar  slippage;     // valor do tamanho do slippage admissível - 1 byte
   char   reserved1;    // pula 1 byte
   short  reserved2;    // pula 2 bytes
   int    reserved4;    // outros 4 bytes são pulados. garantir um alinhamento de 8 by
   double take;         // valores do preço de fixação de lucro
   double stop;         // valor do preço de stop de proteção
  };
```

Tal descrição de estruturas alinhadas é necessário somente para transferência de dados para funções de dll importadas.

Atenção: Este exemplo ilustra dados incorretamente esquematizados. Seria melhor primeiro declarar o *take* e *stop* de grandes volumes de dados do tipo double, e então declarar o membro *slippage* do tipo uchar. Neste caso, a representação interna de dados será sempre a mesma, independentemente do valor especificado no *#pragma pack()*.

Se a estrutura contém variáveis do tipo string e/ou objeto de um array dinâmico, o compilador atribui um construtor implícito para tal estrutura. Este construtor redefine todos os membros de tipo string da estrutura e corretamente inicializa objetos do array dinâmico.

Estruturas Simples

© 2000-2024, MetaQuotes Ltd.

As estruturas contendo cadeias de caracteres, objetos de classe, ponteiros e objetos de matrizes dinâmicas são chamadas de estruturas simples. As variáveis de estruturas simples e suas matrizes podem ser transferidas como parâmetros para as funções importadas a partir de DLL.

Copiar e colar estruturas simples é permitido apenas em dois casos:

- se os objetos pertencem ao mesmo tipo de estrutura
- se os objetos são ligados por uma linha herança, isto é, uma estrutura é um descendente de uma outra estrutura.

Mostraremos isso com ajuda de exemplos, criamos uma estrutura personalizada CustomMqlTick idêntica em composição à estrutura construída MqlTick. O compilador não permitirá tentativas de copiar e colar os valores do objeto MqlTick no objeto de tipo CustomMqlTick. A conversão direta para o tipo desejado também causará um erro de compilação:

```
      //--- copiar estruturas simples de diferentes tipos é restrito
      my_tick1=last_tick;               // aqui o compilador gerará um erro

      //--- combinar estruturas de diferente tipo também é restrito
      my_tick1=(CustomMqlTick)last_tick;// aqui o compilador gerará um erro
```

Portanto, resta só uma opção, isto é, copiar e colar os valores dos membros da estrutura elemento por elemento. Mas, ao fazer isto, é permitido copiar e colar os valores dos objetos do mesmo tipo CustomMqlTick.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- também é possível copiar e colar objetos da mesma estrutura CustomMqlTick
      my_tick2=my_tick1;

      //--- criamos uma matriz a partir de objetos da estrutura simples CustomMqlTick
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

Como um teste é chamada a função ArrayPrint() para exibir os valores da matriz *arr[], no diário*.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criamos a mesma estrutura como a construída por MqlTick
   struct CustomMqlTick
     {
      datetime          time;          // Tempo da última atualização dos preços
      double            bid;           // Preço atual Bid
      double            ask;           // Preço atual Ask
      double            last;          // Preço atual da última transação (Last)
      ulong             volume;        // Volume para o preço atual Last
      long              time_msc;      // Hora da última atualização dos preços em mi
      uint              flags;         // Sinalizadores de ticks
     };
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- obtemos os valores do último ticks
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- tentamos colar e copiar os dados a partir do MqlTick no CustomMqlTick
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- copiar e colar estruturas simples não aparentadas é restrito
      //1. my_tick1=last_tick;               // aqui o compilador gerará um erro

      //--- combinar estruturas não aparentadas também é restrito
      //2. my_tick1=(CustomMqlTick)last_tick;// aqui o compilador gerará um erro

      //--- por isso copiamos e colamos os membros da estrutura elemento por elemento
      my_tick1.time=last_tick.time;
      my_tick1.bid=last_tick.bid;
      my_tick1.ask=last_tick.ask;
      my_tick1.volume=last_tick.volume;
      my_tick1.time_msc=last_tick.time_msc;
      my_tick1.flags=last_tick.flags;

      //--- também é possível copiar e colar objetos da mesma estrutura CustomMqlTick
      my_tick2=my_tick1;

      //--- criamos uma matriz a partir de objetos da estrutura simples CustomMqlTick
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- exemplo de exibição de valores de matriz contendo objetos do tipo CustomMqlTick
      /*
                       [time]    [bid]    [ask]    [last] [volume]     [time_msc] [flags]
      [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      */
     }
   else
      Print("SymbolInfoTick() failed, error = ",GetLastError());
  }
```

O segundo exemplo mostra a possibilidade de copiar e colar estruturas simples segundo linha de herança. Assumamos que temos uma estrutura básica Animal, da qual são geradas - para herança - as estruturas Cat e Dog. Nós podemos copiar e colar entre si mesmos os objetos Animal e Cat (ou Animal e Dog), no entanto não podemos copiar e colar entre sim mesmos Cat e Dog, embora ambos sejam descendentes da estrutura Animal.

```
//--- estrutura para descrever cães
struct Dog: Animal
  {
   bool              hunting;       // raça de caça
```

© 2000-2024, MetaQuotes Ltd.

```
  };
//--- estrutura para descrição de gatos
struct Cat: Animal
  {
   bool              home;          // raça domestica
  };
//--- criamos os objetos das subclasses
   Dog dog;
   Cat cat;
//--- é possível copiar no ancestral e colar no descendente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // cães sabem nadar
//--- é impossível copiar e colar objetos de subestruturas (Dog != Cat)
   cat=dog;        // aqui o compilador gerará um erro
```

Código completo de exemplo:

```
//--- estrutura básica para descrever animais
struct Animal
  {
   int               head;          // número de cabeças
   int               legs;          // número de patas
   int               wings;         // número de assas
   bool              tail;          // presença de cauda
   bool              fly;           // voa
   bool              swim;          // nada
   bool              run;           // corre
  };
//--- estrutura para descrever cães
struct Dog: Animal
  {
   bool              hunting;       // raça de caça
  };
//--- estrutura para descrição de gatos
struct Cat: Animal
  {
   bool              home;          // raça domestica
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criamos um objeto do tipo básico Animal e descrevemo-lo
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
```

© 2000-2024, MetaQuotes Ltd.

```
   some_animal.fly=false;
   some_animal.swim=false;
   some_animal.run=true;
//--- criamos os objetos dos subtipos
   Dog dog;
   Cat cat;
//--- é possível copiar no ancestral e colar no descendente (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // cães sabem nadar
//--- é impossível copiar e colar objetos de subestruturas (Dog != Cat)
   //cat=dog;        // aqui o compilador gerará um erro
//--- por isso é possível copiar e colar apenas elemento por elemento
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // gatos não sabem nadar
//--- é possível copiar valores no descendente e colá-los no ancestral
   Animal elephant;
   elephant=cat;
   elephant.run=false;// elefantes não sabem correr
   elephant.swim=true;// elefantes nadam
//--- criamos uma matriz
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- imprimimos
   ArrayPrint(animals);
//--- resultado da execução
/*
       [head] [legs] [wings] [tail] [fly] [swim] [run]
   [0]      1      4       0   true false  false  true
   [1]      1      4       0   true false   true  true
   [2]      1      4       0   true false  false false
   [3]      1      4       0   true false   true false
*/
  }
```

Outro método para copiar e colar tipos simples consiste em utilizar associações, para fazer isto, os objetos destas estruturas devem ser membros da mesma associação - veja o exemplo em union.

Acesso a Membros de Estrutura

A estrutura é um novo tipo de dados permitindo declarar variáveis deste tipo. A estrutura pode ser declarado somente um vez dentro de um projeto. Os membros de estrutura são acessados usando a operação ponto (.).

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
struct trade_settings
  {
   double take;          // valor do preço de fixação do lucro
   double stop;          // valor do preço stop de proteção
   uchar  slippage;      // valor do slippage admissível
  };
//--- cria e inicializa uma variável do tipo trade_settings
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

pack para alinhamento de campos de estruturas e classes

O atributo especial pack permite definir o alinhamento dos campos de estrutura ou classe.

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

onde n é um dos seguintes valores 1,2,4,8 ou 16. Pode estar ausente.

Exemplo:

```
   struct pack(sizeof(long)) MyStruct
     {
      // membros da estrutura serão alinhados a 8 bytes
     };
ou
   struct MyStruct pack(sizeof(long))
     {
      // membros da estrutura serão alinhados a 8 bytes
     };
```

Por padrão, pack(1) é usado para estruturas. Isso significa que na memória os membros da estrutura estão localizados um após o outro e o tamanho da estrutura é igual à soma dos tamanhos de seus membros.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estrutura simples sem alinhamento
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- declaramos uma instância de uma estrutura simples
```

© 2000-2024, MetaQuotes Ltd.

```
   Simple_Structure s;
//--- exibimos o tamanho de cada membro da estrutura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- verificamos de que o tamanho da estrutura POD é igual à soma dos tamanhos de seu
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Resultado:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

O alinhamento dos campos da estrutura pode ser necessário ao trocar dados com bibliotecas de terceiros (*.DLL), nas quais é aplicado tal alinhamento.

Vamos exemplificar como funciona o alinhamento. Peguemos a estrutura de quatro membros sem alinhamento.

```
//--- estrutura simples sem alinhamento
   struct Simple_Structure pack() // tamanho não especificado, ele será alinhado a 1 k
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- declaramos uma instância de uma estrutura simples
   Simple_Structure s;
```

Os campos da estrutura serão localizados na memória um após o outro, de acordo com a ordem de declaração e o tamanho do tipo. Tamanho da estrutura igual 15, nos arrays será indefinido o deslocamento para os campos da estrutura.

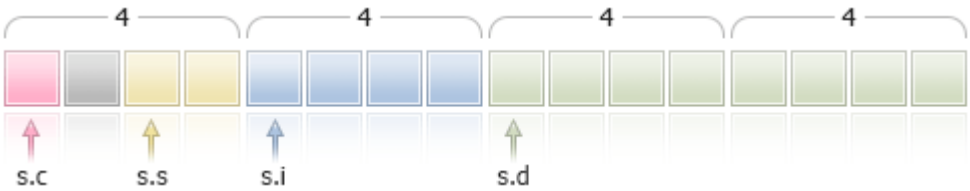
Agora declaramos a mesma estrutura com um alinhamento de 4 bytes e executamos o código.

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

© 2000-2024, MetaQuotes Ltd.

```
//--- estrutura simples com alinhamento de 4 bytes
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- declaramos uma instância de uma estrutura simples
   Simple_Structure s;
//--- exibimos o tamanho de cada membro da estrutura
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- verificamos de que o tamanho da estrutura POD agora não seja igual à soma dos ta
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Resultado:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // tamanho da estrutura alterado
*/
  }
```

O tamanho da estrutura foi alterado para que todos os membros de 4 bytes ou mais tivessem um deslocamento múltiplo de 4 bytes a partir do início da estrutura. Membros menores serão alinhados ao seu próprio tamanho (por exemplo, 2 para short). Veja que o byte adicionado é mostrado em cinza.



Neste caso, após o membro **s.c** é adicionado 1 byte, para que o campo **s.s** (sizeof(short)==2) tenha um limite de 2 byte (alinhamento para o tipo short).

O deslocamento para o início da estrutura no array também será alinhado a 4 bytes, ou seja, para Simple_Structure arr[], os endereços dos elementos a[0], a[1], a[n] serão múltiplos de 4 bytes.

Consideremos mais duas estruturas que consistem no mesmo tipo com alinhamento a 4 bytes, mas cuja ordem de membros é diferente. Na primeira estrutura, os membros são organizados em ordem ascendente segundo o tamanho do tipo.

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

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- estrutura simples com alinhamento a 4 bytes
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- declaramos uma instância de uma estrutura simples
   CharShortInt ch_sh_in;
//--- exibimos o tamanho de cada membro da estrutura
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- verificamos de que o tamanho da estrutura POD é igual à soma dos tamanhos de seu
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  Resultado:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

Como se pode ver, o tamanho da estrutura é 8 e consiste em dois blocos de 4 bytes cada. No primeiro bloco, são colocados os campos com tipos char e short, no segundo - o campo com tipo int.

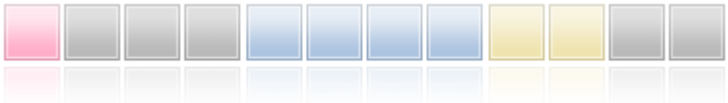
Agora, a partir da primeira estrutura, vamos fazer a segunda, que difere apenas na ordem dos campos: reorganizamos o tipo de membro short no final.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estrutura simples com alinhamento a 4 bytes
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- declaramos uma instância de uma estrutura simples
```

© 2000-2024, MetaQuotes Ltd.

```
   CharIntShort ch_in_sh;
//--- exibimos o tamanho de cada membro da estrutura
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- verificamos de que o tamanho da estrutura POD é igual à soma dos tamanhos de seu
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  Resultado:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
*/
  }
```

Embora a estrutura em si não tenha mudado, a alteração na ordem dos membros levou a um aumento no tamanho da própria estrutura.



Durante a herança, também é preciso levar em conta o alinhamento. Exemplificamos uma estrutura Parent simples, que possui apenas um membro do tipo char. O tamanho dessa estrutura sem alinhamento é 1.

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

Crie uma classe filha Children com adição de um membro do tipo short (sizeof(short)=2).

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

Como resultado, ao definir o alinhamento como 2 bytes, o tamanho da estrutura será 4, embora o tamanho dos próprios membros seja 3. Já que sob a classe pai, para Parent serão alocados 2 bytes, de forma que o acesso ao campo short da classe filha seja alinhado a 2 bytes.

O conhecimento de como a memória é alocada para membros de uma estrutura é necessário se o programa MQL5 interage com dados de terceiros escrevendo/lendo no nível do arquivo ou da thread.

A Biblioteca padrão contém funções para trabalhar com funções WinAPI, no catálogo MQL5\Include\WinAPI. Estas funções usam estruturas com um determinado alinhamento para os casos em que são necessárias para trabalhar com WinAPI.

offsetof é um comando especial que está diretamente relacionado ao atributo pack. Ele permite obter o deslocamento do membro a partir do início da estrutura.

© 2000-2024, MetaQuotes Ltd.

```
//--- declaramos uma variável do tipo Children
   Children child;
//--- encontramos o deslocamento a partir do início da estrutura
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  Resultado:
   offsetof(Children,c)=0
  offsetof(child.s)=2
*/
```

Especificador final

A presença do especificador final, ao declarar a estrutura, proíbe a herança a partir dela. Se a estrutura não precisar de alterações futuras ou se essas alterações não se puderem levar a cabo por causa de questões de segurança, declare-a usando o especificador final. Além disso, todos os membros da estrutura também serão implicitamente considerados como final.

```
struct settings final
  {
  //--- corpo da estrutura
  };

struct trade_settings : public settings
  {
  //--- corpo da estrutura
  };
```

Como no exemplo acima, ao tentar herdar a partir da estrutura, usando o modificador final, o compilador irá emitir um erro:

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

Classes

As classes diferem das estruturas no seguinte:

- a palavra-chave class é usado na declaração;
- por default (padrão), todos os membros da classe têm especificador de acesso private, a menos que seja indicado o contrário. Dados-membros da estrutura têm o tipo default de acesso como public, a menos que seja indicado o contrário;
- objetos de classe sempre tem uma tabela de funções virtuais, mesmo que não existam funções virtuais declaradas na classe. Estruturas não podem ter funções virtuais;
- o operador new pode ser aplicado a objetos de classe; o operador this não pode ser aplicado a estruturas;
- classes pode ser herdados somente de classes, estruturas pode ser herdados somente de estruturas.

Classes e estruturas podem ter um construtor e destrutor explícitos. Se seu construtor for definido explicitamente, a inicialização de uma variável de estrutura ou classe usando a seqüência de inicialização é impossível.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
struct trade_settings
  {
   double take;          // valor do preço de fixação do lucro
   double stop;          // valor do preço stop de proteção
   uchar  slippage;      // valor do slippage admissível
   //--- Construtor
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- Destrutor
         ~trade_settings() { Print("Este é o final"); }
  };
//--- Compilador gerará uma mensagem de erro de que a inicialização é impossível
trade_settings my_set={0.0,0.0,5};
```

Construtores e Destrutores

Um construtor é uma função especial, que é chamada automaticamente ao se criar um objeto de uma estrutura ou classe e normalmente é usado para inicializar membros da classe. Mais adiante, falaremos sobre classes, tudo que for dito se aplica também a estruturas, a menos que seja indicado o contrário. O nome de um construtor deve corresponder ao nome da classe. O construtor não tem tipo de retorno (você pode especificar o tipo void).

Membros de classe definidos - strings, arrays dinâmicos e objetos que exigem inicialização - serão inicializados em qualquer caso,independentemente de haver ou não um construtor.

Cada classe pode ter múltiplos construtores, diferindo pelo número de parâmetros e a lista de inicialização. Um construtor que exige especificação de parâmetros é chamado um construtor paramétrico.

Um construtor sem parâmetros é chamado **um construtor default**. Se nenhum construtor for declarado em uma classe, o compilador cria um construtor default durante a compilação.

```
//+------------------------------------------------------------------+
//| Uma classe para trabalhar com uma data                           |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // Ano
   int               m_month;         // Mês
   int               m_day;           // Dia do mês
   int               m_hour;          // Hora no dia
   int               m_minute;        // Minutos
   int               m_second;        // Segundos
public:
   //--- Construtor default
                     MyDateClass(void);
   //--- Construtor paramétrico
                     MyDateClass(int h,int m,int s);
  };
```

© 2000-2024, MetaQuotes Ltd.

Um construtor pode ser declarado na descrição da classe e então seu corpo pode ser definido. Por exemplo, dois construtores de MyDateClasse podem ser definidos da seguinte maneira:

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

No construtor default, todos os membros da classes são preenchidos usando a função TimeCurrent(), no construtor paramétrico somente os valores de hora são preenchidos. Outros membros da classe (m_year, m_month and m_day) serão inicializados automaticamente com a data corrente.

O construtor default tem um propósito especial ao inicializar um array de objetos de sua classe. Um construtor, o qual todos o parâmetros têm valores default, **não é** um construtor default. Aqui está um exemplo:

```
//+------------------------------------------------------------------+
//| Uma classe com um construtor default                             |
//+------------------------------------------------------------------+
class CFoo
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime          m_call_time;     // Hora da última chamada ao objeto
public:
   //--- Um construtor com um parâmetro que tem um valor default não é um construtor d
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- Um construtor copiador
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo; // Esta variação não pode ser utilizada - um construtor default não foi d
//--- Possíveis opções para criar o objeto CFoo
   CFoo foo1(TimeCurrent());     // Uma explicita chamada de um construtor paramétrico
   CFoo foo2();                  // Uma explícita chamada de um construtor paramétrico
   CFoo foo3=D'2009.09.09';      // Uma implícita chamada de um construtor paramétrico
   CFoo foo40(foo1);             // Uma explicita chamada de um construtor copiador
   CFoo foo41=foo1;              // Uma implícita chamada de um construtor copiador
   CFoo foo5;                    // Uma explícita chamada de um construtor default (se
                                 // então um construtor paramétrico com um valor defau
//--- Possíveis opções para criar o objeto CFoo
   CFoo *pfoo6=new CFoo();       // Criação dinâmica de um objeto e recepção de um por
   CFoo *pfoo7=new CFoo(TimeCurrent());// Outra opções de criação dinâmica de objeto
   CFoo *pfoo8=GetPointer(foo1); // Agora pfoo8 aponta para o objeto foo1
   CFoo *pfoo9=pfoo7;            // pfoo9 e pfoo7 apontam para o mesmo objeto
   // CFoo foo_array[3];         // Esta opção não pode ser usado - um construtor defa
//--- Mostra os valores de m_call_time
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- Exclui dinamicamente arrays criados
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // Você não precisa excluir pfoo8 explicitamente, já que ele apont
   //delete pfoo9;  // Você não precisa excluir pfoo9 explicitamente, já que ele apont
  }
```

Se você descomentar estas strings

```
  //CFoo foo_array[3];     // Esta variante não pode ser usada - um construtor default
```

ou

```
  //CFoo foo_dyn_array[];  // Esta variante não pode ser usada - um construtor default
```

então o compilar retorná um erro para eles "default constructor is not defined" (construtor default não definido).

Se uma classe tiver um construtor definido pelo usuário, o construtor padrão não é gerado pelo compilador. Isso significa que se um construtor paramétrico é declarado em uma classe, mas um construtor default não é declarado, você não pode declarar arrays de objetos desta classe. O compilador retorná um erro para este script:

```
//+------------------------------------------------------------------+
//| Uma classe sem um construtor default                             |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Obtém o erro "default constructor is not defined" durante compilação
   CFoo badFoo[5];
  }
```

Neste exemplo, a classe CFoo tem um construtor paramétrico declarado - em tais casos, o compilador não cria um construtor default automaticamente durante compilação. Ao mesmo tempo quando você declara um array de objetos, presume-se que todos os objetos devam se criados e inicializados automaticamente. Durante a auto-inicialização de um objeto, é necessário chamar um construtor default, mas já que o construtor default não foi explicitamente declarado e nem automaticamente gerado pelo compilador, é impossível criar tal objeto. Por esta razão, o compilador gerou um error na etapa de compilação.

Existe uma sintaxe especial para inicializar um objeto usando um construtor. Inicializadores de membros de uma estrutura ou classe (construções especiais para inicialização) podem ser especificados na lista de inicialização.

Uma lista de inicialização é uma lista de inicializadores separados por vírgulas, que seguem depois do dois pontos (:), depois da lista de parâmetros de um construtor e precede o corpo (antes da abertura de chave). Há vários requisitos:

- Listas de inicialização podem ser usados somente em construtores;
- Membros paternos não podem ser inicializados na lista de inicialização;
- A lista de inicialização deve ser seguida por uma definição (implementação) de uma função.

Aqui está um exemplo de vários construtores para inicializações de membros de uma classe.

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Uma classe para armazenar o nome de um caractere                 |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // Primeiro nome
   string            m_second_name;    // Segundo nome
public:
   //--- Um construtor default vazio
                     CPerson() {Print(__FUNCTION__);};
   //--- Um construtor paramétrico
                     CPerson(string full_name);
   //--- Um construtor com uma lista de inicialização
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Obtém o erro "default constructor is not defined"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                        // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");              // Huckleberry Finn
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // Winnie the Pooh
   //--- Valores de sáida
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- Apaga um objeto criado dinamicamente
   delete Pooh;
  }
```

Neste caso, a classe CPerson tem três construtores:

1. Um construtor default explícito, que permite a criação de um array de objetos desta classe;

© 2000-2024, MetaQuotes Ltd.

2. Um construtor com um parâmetro, que recebe um nome completo como um parâmetro e divide com o nome e segundo nome de acordo com o espaço encontrado;
3. Um construtor com dois parâmetros que contem uma lista de inicialização. Inicializadores - m_second_name(surname) e m_first_name(name).

Note que a inicialização usando uma lista substitui uma atribuição. Membros individuais devem ser inicializados como:

```
class_member (uma lista de expressões)
```

Na lista de inicialização, membros podem aparecer em qualquer ordem, mas todos os membros da classe serão inicializados de acordo com a ordem de sua aparição. Isso significa que no terceiro construtor, primeiro o membro m_first_name será inicializado, já que ele aparece primeiro, e somente depois m_second_name será inicializado. Isto será tomado em conta nos casos onde a inicialização de alguns membros da classe depende dos valores em outros membros da classe.

Se um construtor default não for declarado na base base, e ao mesmo tempo um ou mais construtores com parâmetros forem declarados, você deve sempre chamar um dos construtores da classe base na lista de inicialização. Ele é passado com vírgula da mesma forma que membros comuns da lista e será chamado primeiro durante a inicialização do objeto, não importando onde ele é colocado na lista de inicialização.

```
//+------------------------------------------------------------------+
//| A classe base                                                    |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- Um construtor com uma lista de inicialização
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| Uma classe derivada a partir de CFoo                             |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      // Um membro de classe é um objeto do pai
public:
   //--- O construtor default na lista de inicialização chama o construtor do pai
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

© 2000-2024, MetaQuotes Ltd.

Neste exemplo, ao criar o objeto bar, um construtor default CBar() será chamado, no qual primeiro um construtor do pai de CFoo é chamado, e então vem um construtor para o membro de classe m_member.

Um destrutor é uma função especial que é chamado automaticamente quando um objeto de classe é destruído. O nome do destrutor é escrito como nome de classe precedido de (~). Strings, arrays dinâmicos e objetos, exigindo desinicialização, serão desinicializados de qualquer forma, não importando se o destrutor estiver presente ou ausente. Se existir um destrutor, essas ações serão executadas após chamar o destrutor.

Destrutores são sempre virtuais, não importando se eles são declarados com a palavra-chave virtual ou não.

Definindo Métodos de Classe

Métodos de funções de classe podem ser definidos tanto dentro quanto fora da declaração de classe. Se o método é definido dentro de uma classe, então seu corpo vem logo após a declaração do método.

Exemplo:

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

© 2000-2024, MetaQuotes Ltd.

Funções a partir de SetRightBorder(int border) até Draw() são declarados e definidos diretamente dentro da classe CTetrisShape.

O construtor CTetrisShape() e os métodos CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) e CheckRight(int& side_row[]) são declarados somente dentro da classe, mas ainda não definidos. As definições destas funções serão feitas mais adiante no código. A fim de definir o método do lado de fora da classe, o operador de resolução de scope é usado, o nome da classe é usado como o escopo.

Exemplo:

```
//+------------------------------------------------------------------+
//| Construtor da classe básica                                      |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------+
//| Verificação da capacidade de move para baixo (para a varra e cubo) |
//+------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

Especificadores de Acesso Public, Protected e Private

Quando desenvolver um nova classe, é recomendável restringir o acesso do lado de fora aos membros da classe. São usadas palavras-chave private ou protected para esta finalidade. Neste caso, dados escondidos podem ser acessados somente a partir de métodos-funções da mesma classe. Se a palavra-chave *protected* é usada, dados escondidos podem ser acessados também a partir de métodos de classes herdeiras desta classe. O mesmo método pode ser usado para restringir o acesso de métodos-funções de uma classe.

Se você precisar de acesso com abertura completa aos membros e/ou métodos de uma classe, use a palavra-chave public.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
class CTetrisField
  {
private:
   int              m_score;                          // Contagem
   int              m_ypos;                           // Posição corrente das figu
   int              m_field[FIELD_HEIGHT][FIELD_WIDTH]; // Boa matriz
   int              m_rows[FIELD_HEIGHT];             // Numeração das linhas boas
   int              m_last_row;                       // Última linha livre
   CTetrisShape     *m_shape;                         // Figura Tetris
   bool             m_bover;                          // Fim de jogo
public:
   void             CTetrisField() { m_shape=NULL; m_bover=false; }
   void             Init();
   void             Deinit();
   void             Down();
   void             Left();
   void             Right();
   void             Rotate();
   void             Drop();
private:
   void             NewShape();
   void             CheckAndDeleteRows();
   void             LabelOver();
  };
```

Quaisquer membros de classe e métodos declarados após o especificador **public:** (e antes do próximo especificador de acesso) ficam disponíveis para qualquer referência ao objeto da classe pelo programa. Neste exemplo, existem os seguintes membros: funções CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() e Drop().

Quaisquer membros de classe e métodos declarados após o especificador **private:** (e antes do próximo especificador de acesso) ficam disponíveis somente para as funções-membros desta classe. Especificadores de acesso a elementos sempre terminam com um dois pontos (:) e podem aparecer na definição da classe várias vezes.

Qualquer membro da classe declarado após o especificador de acesso **protected:** (e antes do próximo especificador de acesso) está disponível apenas para funções-membro dessa classe e funções-membro dos herdeiros desta classe. Ao tentar usar membros com especificadores private e protected um erro será recebido de fora no estágio de compilação. Exemplo:

```
class A
  {
protected:
   //--- operador de cópia está disponível apenas na classe A e seus herdeiros
   void operator=(const A &)
     {
     }
  };
class B
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   //--- declarado objeto de classe A
   A                 a;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- declaradas duas variáveis do tipo B
   B b1, b2;
   //--- tentativa de copiar um objeto para outro
   b2=b1;
  }
```

Ao compilar esse código, aparecerá uma mensagem de erro — tentativa de chamar o operador de cópia remoto:

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

A segunda linha abaixo fornece uma descrição mais detalhada — o operador de cópia na classe B foi excluído explicitamente, pois foi chamado o operador de cópia indisponível da classe A:

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

O acesso aos membros de uma classe base pode ser redefinido durante a herança nas classes derivadas.

Especificador delete

O especificador delete marca funções-membro da classe que não podem ser usadas. Isso significa que se o programa se referir explícita ou implicitamente a essa função, aparecerá um erro no estágio de compilação. Por exemplo, esse especificador permite tornar indisponíveis os métodos pai em uma classe filho. Você pode obter o mesmo resultado se declarar na classe pai uma função na área privada (declarações na seção private) Use delete, nesse caso, torna o código mais legível e gerenciável no nível dos herdeiros.

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

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos uma variável do tipo A
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- tentativa de obter um valor de uma variável do tipo B
   B b;
   Print("b.GetValue()=", b.GetValue()); // o compilador dá um erro nessa linha
  }
```

Mensagem do compilador:

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

Com a ajuda do especificador delete, é possível desativar a conversão automática ou o construtor de cópia, que, caso contrário, teria que estar oculto nas seções private. Exemplo:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
   //--- proibimos a chamado com o tipo int
   void              SetValue(int) = delete;
   //--- proibimos o operador de cópia
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos duas variáveis do tipo A
   A a1, a2;
   a1.SetValue(3);       // erro!
   a1.SetValue(3.14);    // OK
   a2=a1;                // erro!
  }
```

Ao tentar compilar, obtemos as mensagens de erro:

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

Especificador final

© 2000-2024, MetaQuotes Ltd.

A presença do especificador final, ao declarar a classe, proíbe a herança a partir dela. Se a interface da classe não precisar de alterações futuras ou se essas alterações não se puderem levar a cabo por causa de questões de segurança, declare-a usando o modificador final. Além disso, todos os métodos da classe também serão implicitamente considerados como final.

```
class CFoo final
  {
  //--- corpo da classe
  };

class CBar : public CFoo
  {
  //--- corpo da classe
  };
```

Como no exemplo acima, ao tentar herdar a partir da classe, usando o especificador final, o compilador irá emitir um erro:

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

Uniões (union)

A união é um tipo de dados especial que compreende uma série de variáveis que compartilham o mesmo espaço de memória. Assim, a união permite interpretar a mesma sequência de bit de dois (ou mais) formas diferentes. A declaração de união é semelhante a declarar estruturas e começa com a palavra-chave union.

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

Mas ao contrário das estruturas, diferentes membros de união pertencem ao mesmo local de memória. Neste exemplo, é declarada a união LongDouble, nela o tipo long e o valor do tipo double compartilham a mesma região memória. É importante entender que é impossível fazer com que esta associação armazene ao mesmo tempo um valor inteiro *long* e um verdadeiro *double* (como acontecia na estrutura), porque as variáveis long_value e double_value são sobrepostas (na memória). Mas o programa MQL5 a qualquer momento pode processar a informação contida nesta união, como um valor inteiro (long) ou real (double). Assim, a união faz com que seja possível obter duas (ou mais) variantes de representação da mesma série de dados.

Ao declarar a união, o compilador automaticamente aloca uma área de memória suficiente para armazenar na união as maiores variáveis de acordo com o volume do tipo. Para acessar o elemento de união, usa-se a mesma sintaxe como para as estruturas, isto é, operador ponto.

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

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- obtemos o número inválido -nan(ind) e exibimo-lo
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- maior número normalizado (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- menor número positivo normalizado (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  Resultado da execução
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Como as uniões permitem que o programa interprete os mesmos dados na memória de diferentes maneiras, elas são frequentemente utilizadas em aplicações que requerem uma conversão de tipos pouco comum.

As uniões não podem participar na herança, além disso, não podem ter membros estáticos por definição. De resto *union* comporta-se como uma estrutura, em que todos os membros têm deslocamento zero. Ao mesmo tempo, os membros de união não podem ser dos seguintes tipos:

- arrays dinâmicas
- strings
- ponteiros de objetos e funções
- classes de objetos
- estruturas de objeto que têm construtores ou destruidores
- estruturas de objeto com um membro a partir dos pontos 1-5

Como as classes, a união pode ter construtores e destruidores, bem como métodos. Por padrão, os membros da união têm o tipo de acesso public<t2, para criar um elementos fechados necessárias para usar a palavra-chave private. Todos esses recursos são mostrados no exemplo que mostra como converter a cor do tipo color, no formato ARGB, como o faz a função ColorToARGB().

```
//+------------------------------------------------------------------+
//| União para conversão de color(BGR) no formato ARGB      |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- construtores
                     ARGB(color col,uchar a=0){Color(col,a);};
```

```
                    ~ARGB(){};
   //--- métodos públicos
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- métodos fechados
private:
   //+-----------------------------------------------------------------+
   //| definição de cor e valor do canal-alfa                          |
   //+-----------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- definimos a cor no membro clr
      clr=col;
      //--- definimos o valor do componente Alpha, isto é, o nível de opacidade
      argb[3]=alpha;
      //--- mudamos de lugar os bytes do componente R e B (Red e Blue)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+-----------------------------------------------------------------+
//| Script program start function                                   |
//+-----------------------------------------------------------------+
void OnStart()
  {
//--- o valor 0x55 significa 55/255=21.6 % (0% - totalmente transparente)
   uchar alpha=0x55;
//--- o tipo color têm o formato 0x00BBGGRR
   color test_color=clrDarkOrange;
//--- aqui entrarão os valores de bytes a partir da união ARGB
   uchar argb[];
   PrintFormat("0x%.8X - assim se vê o tipo color para %s, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- o tipo ARGB é apresentado como 0x00RRGGBB, mudados de lugar os componentes RR e
   ARGB argb_color(test_color);
//--- copiamos o array de bytes
   ArrayCopy(argb,argb_color.argb);
//--- vemos como fica no formato ARGB
   PrintFormat("0x%.8X - presentação ARGB com alfa-canal=0x%.2x, ARGB=(%d,%d,%d,%d)",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- adicionamos o valor de opacidade
   argb_color.Alpha(alpha);
//--- tentamos imprimir ARGB como tipo color
   Print("ARGB como color=(",argb_color.clr,")  alfa-canal=",argb_color.Alpha());
//--- copiamos o array de bytes
   ArrayCopy(argb,argb_color.argb);
//--- assim fica no formato ARGB
   PrintFormat("0x%.8X - presentação ARGB com alfa-canal=0x%.2x, ARGB=(%d,%d,%d,%d)",
```

© 2000-2024, MetaQuotes Ltd.

```
                argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- verificamos se é gerada a função ColorToARGB()
   PrintFormat("0x%.8X - resultado ColorToARGB(%s,0x%.2x)",ColorToARGB(test_color,alph
                ColorToString(test_color,true),alpha);
  }
/* Resultado da execução
   0x00008CFF - assim se vê o tipocolor para clrDarkOrange, BGR=(255,140,0)
  0x00FF8C00 - presentação ARGB com alfa-canal=0x00, ARGB=(0,255,140,0)
   ARGB como color=(0,140,255)  alfa-canal=85
  0x55FF8C00 - presentação ARGB com alfa-canal=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - resultado ColorToARGB(clrDarkOrange,0x55)
*/
```

Interfaces

A interface é projetada para determinar a funcionalidade específica que a classe pode implementar. Na verdade, essa é uma classe que não pode conter nenhum membro e não pode ter um construtor e/ou destrutor. Todos os métodos declarados na interface são normalmente virtuais, mesmo sem uma definição explícita.

A interface é definida usando a palavra-chave interface, como é mostrado no exemplo a seguir:

```
//--- interface básica para descrever animais
interface IAnimal
  {
//--- métodos da interface padrão têm acesso público
   void Sound();  // som que produz o animal
  };
//+------------------------------------------------------------------+
//|  a classe CCat é herdada da interface IAnimal                    |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("Cat was born"); }
                    ~CCat() { Print("Cat is dead");  }
   //--- implementamos o método Sound da interface IAnimal
   void Sound(){ Print("meou"); }
  };
//+------------------------------------------------------------------+
//|  a classe CDog é herdada da interface IAnimal                    |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Dog was born"); }
                    ~CDog() { Print("Dog is dead");  }
   //--- implementamos o método Sound da interface IAnimal
   void Sound(){ Print("guaf"); }
```

© 2000-2024, MetaQuotes Ltd.

```
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- matriz do ponteiro para o objeto do tipo IAnimal
   IAnimal *animals[2];
//--- geramos descendente IAnimal e salvamos os ponteiros para eles nas suas matrizes
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- chamamos o método Sound() da interface base IAnimal para cada descendente
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- removemos objetos
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- resultado da execução
/*
   Cat was born
   Dog was born
   meou
   guaf
   Cat is dead
   Dog is dead
*/
  }
```

Como as classes abstratas, não se deve criar o objeto da interface sem descendente. A interface pode ser herdada apenas a partir de outras interfaces e pode se tornar descendente para a classe. Além disso, a interface sempre tem visibilidade pública.

É impossível declarar a interface dentro da declaração da classe ou estrutura, porém é possível armazenar o ponteiro para a interface na variável do tipo void *. De um modo geral, é possível armazenar os ponteiros para objetos de qualquer classe na variável do tipo void *. Para converter um ponteiro void * para outro ponteiro para um objeto de uma classe específica, é indispensável utilizar o operador dynamic_cast. Quando a conversão é impossível, o resultado da operação dynamic_cast será NULL.

Veja Também

Programação Orientada a Objetos

© 2000-2024, MetaQuotes Ltd.

Objeto Array Dinâmico

Arrays Dinâmicos

No máximo pode-se declarar arrays de 4 dimensões. Quando declarar um array dinâmico (um array sem valor especificado no primeiro par de colchetes), o compilador automaticamente cria uma variável da estrutura acima (um objeto array dinâmico) e fornece um código para a correta inicialização.

Arrays dinâmicos são automaticamente liberados quando ficam além da área de visibilidade do bloco em que eles foram declarados.

Exemplo:

```
double matrix[][10][20]; // array dinâmico de 3 dimensões
ArrayResize(matrix,5);   // define o tamanha da primeira dimensão
```

Arrays Estáticos

Quando todas as dimensões significantes do array são explicitamente especificadas, o compilador pré-aloca o tamanho de memória necessário. Tal array é chamado estático. Contudo, o compilador aloca memória adicional para o array dinâmico, que é associado ao buffer estático pré-alocado (parte da memória para armazenar o array).

A criação de um objeto array dinâmico é devido a possível necessidade de passar este array estático como parâmetro para alguma função.

Exemplos:

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

Arrays em Estruturas

Quando um array estático é declarado como membro de uma estrutura, um objeto array dinâmico não é criado. Isso é assim para assegurar compatibilidade com estruturas de dados usados em API's do Windows.

Contudo, arrays estáticos que são declarados como membros de estruturas, podem também serem passados para funções em MQL5. Neste caso, quando passar o parâmetro um objeto temporário de um array dinâmico, conectado com o array estático - membro da estrutura, será criado.

Também Veja

Funções com Arrays, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Matrizes e vetores

Um vector é um tipo de dados especial em MQL5 para trabalhar com vetores, que é uma matriz unidimensional do tipo double. O vetor é um dos conceitos fundamentais da álgebra linear e é usado ativamente em muitos campos da ciência, incluindo física, geometria e outros. Os vetores são usados para resolver sistemas de equações lineares, em gráficos 3D, etc. Com base nos vetores, são definidas operações de adição e multiplicação, e é introduzido o conceito de "Norma" para obter o comprimento ou distância entre vetores. Na programação, os vetores são normalmente representados por matrizes de elementos homogêneos sobre os quais não podem ser realizadas operações vetoriais convencionais, ou seja, as matrizes não podem ser adicionadas umas às outras, multiplicadas umas pelas outras, e não existe o conceito de norma.

Os vetores podem ser representados como um vetor de coluna ou como um vetor de linha quando se trabalha com matrizes. Além disso, em álgebra linear os vetores usam conceitos como covariância e contravariância. Para fins de escrita de código em MQL5, essas diferenças não importam, pois apenas o programador decide qual é cada objeto do tipo vector. Por exemplo, pode ser um vetor de rotação, deslocamento, compressão e assim por diante quando se trata de gráficos 3D.

De um modo geral, do ponto de vista da álgebra linear, um número também é um vetor, só que em um espaço unidimensional. Além disso, um vetor pode ser considerado um caso especial de matriz.

Um matrix é outro tipo de dados especial em MQL5 para representar matrizes, que na verdade é uma matriz bidimensional do tipo double. Vetores e matrizes foram introduzidos na linguagem MQL5 para facilitar o trabalho com certos tipos de conjuntos de dados. Eles permitem que os programadores tirem total proveito da álgebra linear da forma mais simples e próxima da matemática possível. As matrizes são amplamente utilizadas para escrever sistemas compactos de equações algébricas lineares ou diferenciais. Nesse caso, o número de linhas da matriz corresponde ao número de equações e o número de colunas corresponde ao número de incógnitas. Como resultado, a solução de sistemas de equações lineares é reduzida a operações em matrizes.

Existem os seguintes tipos de dados:

- matrix — matriz contendo elementos do tipo double
- matrixf — matriz contendo elementos do tipo float
- matrixc — matriz contendo elementos do tipo complex.
- vector — vetor contendo elementos do tipo double
- vectorf — vetor contendo elementos do tipo float
- vectorc — vetor contendo elementos do tipo complex

Para uso em funções de modelo, é possível usar a entrada matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> em vez dos tipos correspondentes.

Para uma matriz, são definidas as seguintes operações algébricas:

- adição de matrizes do mesmo tamanho;
- multiplicação de matrizes de tamanho adequado, com o número de colunas da matriz à esquerda correspondendo ao número de filas da matriz à direita;
- multiplicação de matrizes por um vetor de coluna e multiplicação de um vetor de linha por uma matriz de acordo com a regra de multiplicação de matrizes. Ou seja, o vetor é, nesse sentido, um caso especial da matriz;
- multiplicação de matriz por um número, ou seja, por um escalar.

© 2000-2024, MetaQuotes Ltd.

No campo da matemática, são considerados muitos tipos e tipos diferentes de matrizes. Por exemplo, identidade, simétrica, assimétrica, triangular superior (triangular inferior) e outras matrizes. De particular importância na teoria da matriz são todos os tipos de formas normais, ou seja, a forma canônica à qual uma matriz pode ser reduzida através da substituição de coordenadas. Na prática, são utilizadas formas normais que possuem propriedades adicionais, tais como estabilidade.

O uso de vetores e matrizes, ou melhor, de métodos especiais que estes tipos de dados possuem, permite escrever um código mais simples, mais curto e mais claro que se aproxima da notação matemática. Isto poupa ao programador o trabalho de criar laços aninhados e de lembrar de indexar corretamente as matrizes que estão envolvidas no cálculo. Isto aumenta a confiabilidade e a velocidade de desenvolvimento de programas complexos.

Lista de métodos implementados em matrix e vector

Os tipos matrix e vector têm métodos incorporados que correspondem às contrapartidas da biblioteca NumPy. Desta forma, você pode migrar algoritmos e códigos de Python para MQL5 com um esforço mínimo. Assim, muitos problemas de processamento de dados, resolução de equações matemáticas, redes neurais e aprendizagem de máquinas podem ser resolvidos usando bibliotecas de linguagem Python prontas para uso.

Método matrix/vector	Contraparte em NumPy	Descrição
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Cria uma matriz com uns na diagonal e zeros nos outros lugares
void matrix.Identity(const int rows)	identity	Cria uma matriz quadrada com uns na diagonal principal
void matrix.Ones(const int rows, const int cols)	ones	Cria uma nova matriz por número de filas e colunas, preenchida com unidades
void matrix.Zeros(const int rows, const int cols)	zeros	Cria uma nova matriz por número de filas e colunas, preenchida com zeros
void matrix.Full(const int rows, const int cols, const scalar value)	full	Cria uma nova matriz pelo número de filas e colunas, preenchidas com um valor escalar
void matrix.Copy(const matrix a)	copy	Cria uma cópia de dada matriz
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	Cria uma matriz a partir de uma matriz unidimensional
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	Cria uma matriz a partir de dados em um arquivo de texto ou binário

© 2000-2024, MetaQuotes Ltd.

Método matrix/vector	Contraparte em NumPy	Descrição
void vector.FromString(const string source, const string sep=" ")	fromstring	Cria um vetor baseado em dados de texto a partir de uma string
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Cria um vetor com valores uniformemente distribuídos em um dado intervalo
void matrix.Diag(const vector v, const int ndiag=0)	diag	Extrai uma diagonal ou cria uma matriz diagonal
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	Cria uma matriz com uns na diagonal indicada e abaixo e zeros nos outros lugares
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	Retorna uma cópia da matriz com elementos zerados sobre a diagonal k
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	Retorna uma cópia da matriz com elementos zerados abaixo da diagonal k
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Cria uma matriz de Vandermonde
vector matrix.Row(const unsigned nrow)		Retorna um vetor de linha
vector matrix.Col(const unsigned ncol)		Retorna um vetor de coluna
unsigned matrix.Rows()		Retorna o número de linhas em uma matriz
unsigned matrix.Cols()		Retorna o número de colunas em uma matriz
void matrix.Init()		Inicializa uma matriz
matrix matrix.Transpose()	transpose	Troca os eixos de uma matriz, retorna a matriz modificada
matrix matrix.Dot(const matrix b)	dot	Produto de duas matrizes
matrix matrix.Inner(const matrix b)	inner	Produto interno de duas matrizes
matrix matrix.Outer(const matrix b)	outer	Produto externo de duas matrizes
matrix matrix.MatMul(const matrix b)	matmul	Produto de matriz de duas matrizes
matrix matrix.MatrixPower(const int power)	matrix_power	Eleva uma matriz quadrada para a potência (inteira) n

© 2000-2024, MetaQuotes Ltd.

Método matrix/vector	Contraparte em NumPy	Descrição
matrix matrix.Kron(const matrix b)	kron	Retorna o produto de Kronecker de duas matrizes
bool matrix.Cholesky(matrix& L)	cholesky	Retorna a decomposição de Cholesky
bool matrix.QR(matrix& Q, matrix& R)	qr	Calcula a fatoração qr de uma matriz
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Calcula a decomposição SVD
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Calcula os valores próprios e os vetores próprios direitos de uma matriz quadrada
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	Retorna os autovalores e autovetores de uma matriz Hermitiana
bool matrix.EigVals(vector& eigen_values)	eigvals	Calcula os valores próprios de uma matriz comum
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Calcula os valores próprios de uma matriz Hermitiana
bool matrix.LU(matrix& L, matrix& U)		Executa a decomposição LU de uma matriz como o produto de uma matriz triangular inferior e uma matriz triangular superior
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		Executa a decomposição LUP de rotação parcial, que é fatoração LU com permutações de linha: PA = LU
double matrix.Norm(const norm)	norm	Retorna a norma de uma matriz ou vetor
double matrix.Norm(const norm)	cond	Calcula o número da condição de uma matriz
vector matrix.Spectrum()		Calcula o espectro de uma matriz como um conjunto de seus autovalores a partir do produto AT*A
double matrix.Det()	det	Calcula o determinante de uma matriz
int matrix.Rank()	matrix_rank	Retorna a posto de matriz usando o método Gaussiano
int matrix.SLogDet(int& sign)	slogdet	Calcula o sinal e o logaritmo de um determinante de matriz

© 2000-2024, MetaQuotes Ltd.

Método matrix/vector	Contraparte em NumPy	Descrição
double matrix.Trace()	trace	Retorna a soma das diagonais de uma matriz
vector matrix.Solve(const vector b)	solve	Resolve uma equação de matriz linear ou um sistema de equações algébricas lineares
vector matrix.LstSq(const vector b)	lstsq	Resolve um sistema de equações algébricas lineares aproximadamente (para matrizes não quadradas ou degeneradas)
matrix matrix.Inv()	inv	Calcula o inverso (multiplicativo) de uma matriz
matrix matrix.PInv()	pinv	Calcula o pseudoinverso pelo método Moore-Penrose
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		Compara os elementos de duas matrizes/vetores com a precisão especificada
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	Permite endereçar um elemento da matriz por meio de um índice, e não por meio de dois
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Retorna o índice do valor máximo
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Retorna o índice do valor mínimo
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Retorna o valor máximo em uma matriz/vetor
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	Calcula a média aritmética dos valores dos elementos
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Retorna o valor mínimo em uma matriz/vetor

© 2000-2024, MetaQuotes Ltd.

Método matrix/vector	Contraparte em NumPy	Descrição
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Executa uma soma de elementos de matriz/vetor, que também pode ser realizada ao longo do(s) eixo(s) especificado(s).
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Limita os elementos de uma matriz/vetor a um intervalo especificado de valores válidos
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Retorna o produto cumulativo (cumulativo) dos elementos da matriz/vetor, incluindo aqueles ao longo do eixo fornecido
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	Retorna a soma cumulativa (cumulativa) de elementos de matriz/vetor, incluindo aqueles ao longo do eixo fornecido
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	Calcula o produto de elementos de matriz/vetor, que também podem ser executados ao longo do eixo especificado
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Altera a forma de uma matriz sem mudar seus dados
void matrix.Resize(const ulong rows,const ulong cols)	resize	Retorna uma nova matriz com a forma especificada
bool matrix.SwapRows(const ulong row1, const ulong row2)		Troca de linhas na matriz
bool matrix.SwapCols(const ulong col1, const ulong col2)		Troca colunas na matriz
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Retorna a faixa de valores de matriz/vetor ou o eixo de matriz especificado e equivalente ao resultado Max() - Min()
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	Calcula o percentil especificado (percentil) dos valores dos elementos de matriz/vetor ou elementos ao longo do eixo especificado. Os valores válidos do parâmetro de porcentagem estão no intervalo [0, 100]

© 2000-2024, MetaQuotes Ltd.

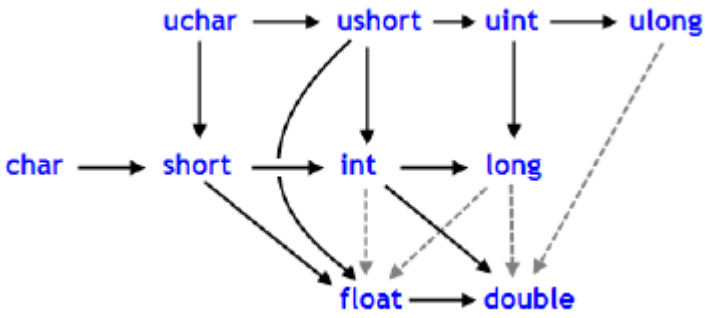
Método matrix/vector	Contraparte em NumPy	Descrição
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Calcula o quantil especificado dos valores dos elementos da matriz/vetor ou dos elementos ao longo do eixo especificado. O intervalo do parâmetro percentual assume valores no intervalo [0, 1]
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Calcula a mediana dos elementos de uma matriz/vetor. A mediana é um valor tal que exatamente metade dos elementos da matriz/vetor será menor que ela, e a outra será maior.
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	Calcula a média ponderada aritmética dos valores dos elementos da matriz/vetor. A soma dos pesos no denominador não pode ser 0, mas alguns pesos podem ser 0
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Calcula o desvio quadrático médio (padrão) dos valores dos elementos de uma matriz/vetor ou um eixo especificado
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Calcula a variância dos valores dos elementos de uma matriz/vetor
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Calcula o coeficiente de correlação de Pearson (coeficiente de correlação linear). O coeficiente de correlação está no intervalo [-1, 1]
vector vector.Correlate(const vector& v,enum mode)	correlate	Calcula os valores da função de correlação cruzada (correlação cruzada) de dois vetores. O parâmetro mode determina o modo de cálculo de convolução linear
vector vector.Convolve(const vector& v, enum mode)	convolve	Retorna uma convolução linear discreta de dois vetores de uma sequência. O parâmetro mode determina o modo de cálculo de convolução linear
matrix matrix.Cov() matrix vector.Cov(const vector& v); (matriz final 2 x 2)	cov	Calcula a matriz de covariância. A covariância de duas amostras (de duas variáveis aleatórias) é uma medida de sua dependência linear

© 2000-2024, MetaQuotes Ltd.

Conversão de Tipo (Typecasting)

Conversão de Tipos Numéricos

Frequentemente surge a necessidade de converter um tipo numérico em outro. Nem todos os tipos numéricos podem ser convertidos em outro. Aqui está o esquema de conversão permitida:



Linhas sólidas com setas indicam mudanças que são realizadas quase sem nenhuma perda de informação. Ao invés do tipo char, o tipo bool pode ser usado (ambos ocupam 1 byte de memória), ao invés do tipo int, o tipo color pode ser usado (4 bytes), ao invés do tipo long, datetime pode ser usado (ocupa 8 bytes). As quatro linhas cinzentas tracejadas, também com setas, denotam conversão em que a perda de precisão pode ocorrer. Por exemplo, o número de algarismos em um inteiro igual a 123456789 (int) é maior que o número de algarismos que podem ser representados por um float.

```
   int n=123456789;
   float f=n;    // o conteúdo de f é igual a 1.234567892E8
   Print("n = ",n,"   f = ",f);
   // resultado n= 123456789    f= 123456792.00000
```

O número convertido em float tem a mesma ordem de grandeza, mas é menos preciso. Conversões, no sentido contrário das setas pretas, podem ser realizadas com possível perda de dados. Conversões entre char e uchar, short e ushort, int e uint, long e ulong (conversões para ambos os lados), podem levar a perda de dados.

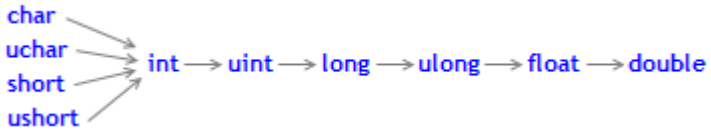
Como resultado da conversão de valores de ponto flutuante para o tipo inteiro, a parte fracionária é sempre excluída. Se você quiser arredondar um float para o inteiro mais próximo (que em muitos casos é mais útil), você deve usar MathRound().

Exemplo:

```
//--- Aceleração gravitacional
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
  Resultado:
   round_g = 9
   math_round_g = 10
*/
```

© 2000-2024, MetaQuotes Ltd.

Se dois valores são combinados por um operador binário, antes da execução da operação, o operando do tipo mais baixo é convertido para o do tipo mais alto, em concordância com a prioridade dada no esquema abaixo:



Os tipos de dados char, uchar, short, e ushort incondicionalmente são convertido para o tipo int.

Exemplos:

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

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

A expressão calculada consiste de duas operações. No primeiro exemplo, a variável c1 do tipo char é convertida para uma variável temporária do tipo int, porque o segundo operando da operação de divisão, a constante 2, é do tipo mais alto, int. Como resultado da divisão de inteiros, 3/2, nós obtemos o valor 1, que é do tipo inteiro.

Na segunda operação do primeiro exemplo, o segundo operando é a constante 0.3, que é do tipo double, então o resultado da primeira operação é convertido em uma variável temporária para o tipo double com o valor 1.0.

No segundo exemplo, a variável c1 do tipo char é convertida para uma variável temporária do tipo double, porque o segundo operando da operação de divisão, a constante 2.0, é do tipo double; nenhuma conversão adicional é feita.

Conversão de Tipos Numéricos

Nas expressões da linguagem MQL5 tanto conversão explicita quanto implícita podem ser usadas. A conversão explícita é escrita da seguinte forma:

```
var_1 = (type)var_2;
```

Uma expressão ou resultado de execução de uma função podem ficar no lugar da variável var_2. A notação funcional de uma conversão explícita é também possível:

```
var_1 = type(var_2);
```

Vamos considerar uma conversão explícita com base no primeiro exemplo.

```
//--- Terceiro exemplo
```

© 2000-2024, MetaQuotes Ltd.

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

Antes da operação de divisão ser realizada, a variável c1 é explicitamente convertida para o tipo double. Agora a constante inteira 2 é convertida no valor 2.0 do tipo double, pois o resultado da conversão do primeiro operando assumiu o tipo double. De fato, a conversão explícita é uma operação unária.

Além disso, quanto tentar converter tipos, o resultado pode ir além da faixa permitida. Neste caso, o truncamento ocorre. Por exemplo:

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

Antes de operações (exceto operações de atribuição) serem realizadas, os dados são convertidos para o tipo de máxima prioridade. Antes de operações de atribuição serem realizadas, os dados são convertidos para o tipo destino.

Exemplos:

```
   int    i=1/2;         // nenhuma conversão, o resultado é 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // a expressão é convertida para o tipo double,
   Print("k = 1/2  ",k);  // então é convertido para o tipo destino, o resultado é 0

   double d=1.0/2.0;     // nenhuma conversão de tipos, o resultado é 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;       // a expressão é convertida para o tipo double,
   Print("e = 1/2.0; ",e);// que é o mesmo do tipo destino, o resultado é 0.5

   double x=1/2;         // a expressão do tipo inteiro é convertido para o tipo destir
   Print("x = 1/2; ",x);  // o resultado é 0.0
```

Ao converter o tipo long/ulong em double, a precisão pode ser perdida caso o valor inteiro seja maior do que 2233720368547747 84 ou menor do que -9223372036854774784.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- definir o valor inteiro maior, que não perde a precisão ao ser lançado para doul
   while(l_max!=long((double)l_max))
      l_max--;
//--- definir o valor inteiro menor, que não perde a precisão ao ser lançado para doul
   while(l_min!=long((double)l_min))
```

© 2000-2024, MetaQuotes Ltd.

```
      l_min++;
//--- resulta no intervalo encontrado para valores inteiro
   PrintFormat("Ao converter um valor inteiro para double, ele deve ser "
               "dentro [%I64d, %I64d] do intervalo",l_min,l_max);
//--- agora, vamos ver o que acontece se o valor fica fora deste intervalo
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
//--- receber o seguinte resultado
// Ao converter um valor integer para double, ele deve estar dentro do intervalo [-922
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

Conversão para o tipo String

O tipo string tem a mais alta prioridade entre os tipos simples. Portanto, se um dos operandos de uma operação é do tipo string, o segundo operando será convertido para uma string automaticamente. Observe que para uma string, uma única operação de dois-lugares diádicos de adição é possível. A conversão explicita de string para qualquer tipo numérico é permitida.

Exemplos:

```
   string s1=1.0/8;            // a expressão é convertida para o tipo double,
   Print("s1 = 1.0/8; ",s1);     //  daí é convertida para o tipo string,
// resultado é "0.12500000" (uma string contém 10 caracteres)

   string s2=NULL;             // string desinicialização
   Print("s2 = NULL; ",s2);      // o resultado é uma string vazia
   string s3="Ticket N"+12345; // a expressão é convertida para o tipo string
   Print("s3 = \"Ticket N\"+12345",s3);

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

Conversão de Ponteiros de Classe Base para Ponteiros de Classes Derivadas

Objetos de classe derivada podem ser vistos como objetos da correspondente classe base. Isso leva a algumas conseqüências interessantes. Por exemplo, apesar do fato que objetos de diferentes classes,

© 2000-2024, MetaQuotes Ltd.

gerados a partir de uma única classe base, poderem diferir significativamente um do outro, podemos criar uma lista ligada (List) com eles, já que vemos eles como objetos do tipo base. Mas o inverso não é verdade: objetos da classe base não são automaticamente objetos da classe derivada.

Você pode usar a conversão explicita para converter ponteiros da classe base para ponteiros da classe derivada. Mas você deve estar totalmente confiante na admissibilidade de tal transformação, porque senão um erro crítico de execução ocorrerá e o programa MQL5 será interrompido.

Conversão dinâmica de tipos usando o operador dynamic_cast

Existe a possibilidade de realizar uma conversão dinâmica de tipos usando o operador dynamic_cast, ele pode ser aplicado apenas para ponteiros de classes. Além disso, a revisão dos tipos é realizada quando o programa está em execução. Isto significa que, ao usar operador compilador dynamic_cast não se verifica o tipo de dados usado para a conversão. Se for levada a cabo conversão do apontador para o tipo de dados que não é um tipo real de objeto, o resultado será o valor NULL.

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

O parâmetro *type-id* em colchetes angulares deve ser o ponteiro um tipo de classe definido anteriormente. O tipo de operando *expression* (ao contrário de C++) pode ser qualquer um, exceto void.

Exemplo:

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

void OnStart()
  {
   CBar bar;
//--- permitida a conversão dinâmica do tipo de ponteiro *bar para o ponteiro *foo
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // não acontecerão erros críticos de execuc
   Print(foo);                             // foo=NULL
//--- restringida a tentativa de clara conversão dos enlaces do objeto do tipo Bar par
   foo=(CFoo *)&bar;                       // acontecerá um erro crítico de execução
   Print(foo);                             // esta linha não será executada
  }
```

Também Veja

Tipos de Dados

© 2000-2024, MetaQuotes Ltd.

Tipo void e constante NULL

Sintaticamente o tipo void é um tipo fundamental, juntamente com os tipos char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double e string. Este tipo é usado tanto para indicar que a função não retorna nenhum valor ou, como um parâmetro de função, para indicar ausência de parâmetros.

A predefinida variável constante **NULL** é do tipo *void*. Ele pode ser atribuído a variáveis de qualquer outro tipo fundamental sem conversão. A comparação de variáveis de tipo fundamental com o valor **NULL** é permitida..

Exemplo:

```
//--- Se a string não é inicializada, então atribua nosso valor predefinido a ela
if(some_string==NULL) some_string="empty";
```

Também **NULL** pode ser comparado com ponteiros de objetos criados com o operador new.

Também Veja

Variáveis, Funções

© 2000-2024, MetaQuotes Ltd.

Tipos personalizados

A palavra-chave typedef em linguagem C++ permite criar tipos personalizados de dados, basta definir o nome do tipo de dados para um tipo de dados existente. Ao fazer isto, não é criado um novo tipo de dados, mas sim é definido um novo nome para o tipo existente. Graças ao uso de tipos personalizados, você pode tornar o programa mais flexível, basta suficiente alterar as instruções typedef usando os macros de substituição (#define). Usar os tipos personalizados também pode melhorar a legibilidade do código, uma vez que, para os tipos de dados padrão, você pode usar seus próprios nomes descritivos com ajuda de typedef. Formato geral para escrever instruções a fim de criar um tipo personalizado:

```
   typedef tipo novo_nome;
```

Aqui o elemento *tipo* representa qualquer tipo de dados válido, enquanto o elemento *novo_nome* é o nome novo para este tipo. É importante notar que o novo nome é definido apenas como um complemento para o nome existente do tipo e não para substituí-lo. Na linguagem MQL5, você pode criar um ponteiro para a função, usando typedef.

Ponteiro para a função

Ponteiro para a função geralmente é determinado pelo formato de registro

```
   typedef tipo_de_resultado_de_função (*Nome_de_tipo_de_função)(lista_de_tipos_de_pa
```

onde, após a palavra typedef, é definida a assinatura da função, isto é, o número e tipo de parâmetros de entrada, bem como o tipo de resultado a ser retornado pela função. Aqui está uma explicação de como criar e usar um ponteiro para uma função:

```
// --- declaramos o ponteiro para uma função que aceita dois parâmetros do tipo int
   typedef int (*TFunc)(int,int);
//--- TFunc é o tipo, e nós podemos declarar o ponteiro-variável para a função
   TFunc func_ptr; // ponteiro para a função
//--- declaramos as funções que correspondem à descrição TFunc
   int sub(int x,int y) { return(x-y); }  // / subtração de um número a partir de out
   int add(int x,int y) { return(x+y); }  // adição de dois números
   int neg(int x)       { return(~x);  }  // inversão de bits na variável
//--- você pode armazenas o endereço da função na variável func_ptr, para chamá-la no
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // erro: neg não tem o tipo int (int,int)
   Print(func_ptr(10));    // erro: deve haver dois parâmetros
```

Neste exemplo, à variável *func_ptr* podem ser atribuídas as funções *sub* e *add*, uma vez que cada uma delas tem dois parâmetros de entrada do tipo int, conforme especificado na definição do ponteiro para a função *TFunc*. Aqui à função *neg* não pode ser atribuído o ponteiro func_ptr, uma vez que sua assinatura é diferente.

Organização dos modelos de eventos na interface personalizada

© 2000-2024, MetaQuotes Ltd.

Usando os ponteiros para a função você pode facilmente construir a manipulação de eventos ao criar a interface personalizada. Mostraremos um exemplo a partir da seção CButton sobre como criar botões e adicionar neles uma função para processamento do carregamento do botão. Em primeiro lugar, definimos o ponteiro para a função *TAction*, ela será chamada pressionando um botão, e criaremos três funções em conformidade com a descrição *TAction*.

```
//--- criamos o tipo personalizado de função
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Abre o arquivo                                                  |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Salva o arquivo                                                 |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Fecha o arquivo                                                 |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Logo realizamos a classe MyButton a partir do CButton, em que adicionamos o membro *TAction* que, por sua vez, é o ponteiro para a função.

```
//+------------------------------------------------------------------+
//| Criamos nossa classe de botão com a função de manipulador de eventos          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // manipulador de eventos para o gra
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- construtor com indicação do texto do botão e ponteiro para a função a fim de
                     MyButton(string text, TAction act)
     {
```

```
      Text(text);
      m_action=act;
     }
   //--- definição de função que será chamada a partir do manipulador de eventos OnEve
   void              SetAction(TAction act){m_action=act;}
   //--- manipulador padrão de eventos de gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL & lparam==Id())
        {
         //--- chamamos o manipulador próprio m_action()
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- retornamos o resultado da chamada do manipulador a partir da classe mão CB
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

Em seguida, criamos a classe derivada CControlsDialog a partir da CAppDialog, à qual adicionamos a matriz *m_buttons* para armazenas os botões do tipo *MyButton*, bem como os métodos *AddButton(MyButton &button)* e *CreateButtons()*.

```
//+------------------------------------------------------------------+
//| Classe CControlsDialog                                           |
//| Designação: painel gráfico para controle do aplicativo           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // matriz de botões
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- adição de botão
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- criação de botões
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Criação do objeto CControlsDialog no gráfico                     |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
```

© 2000-2024, MetaQuotes Ltd.

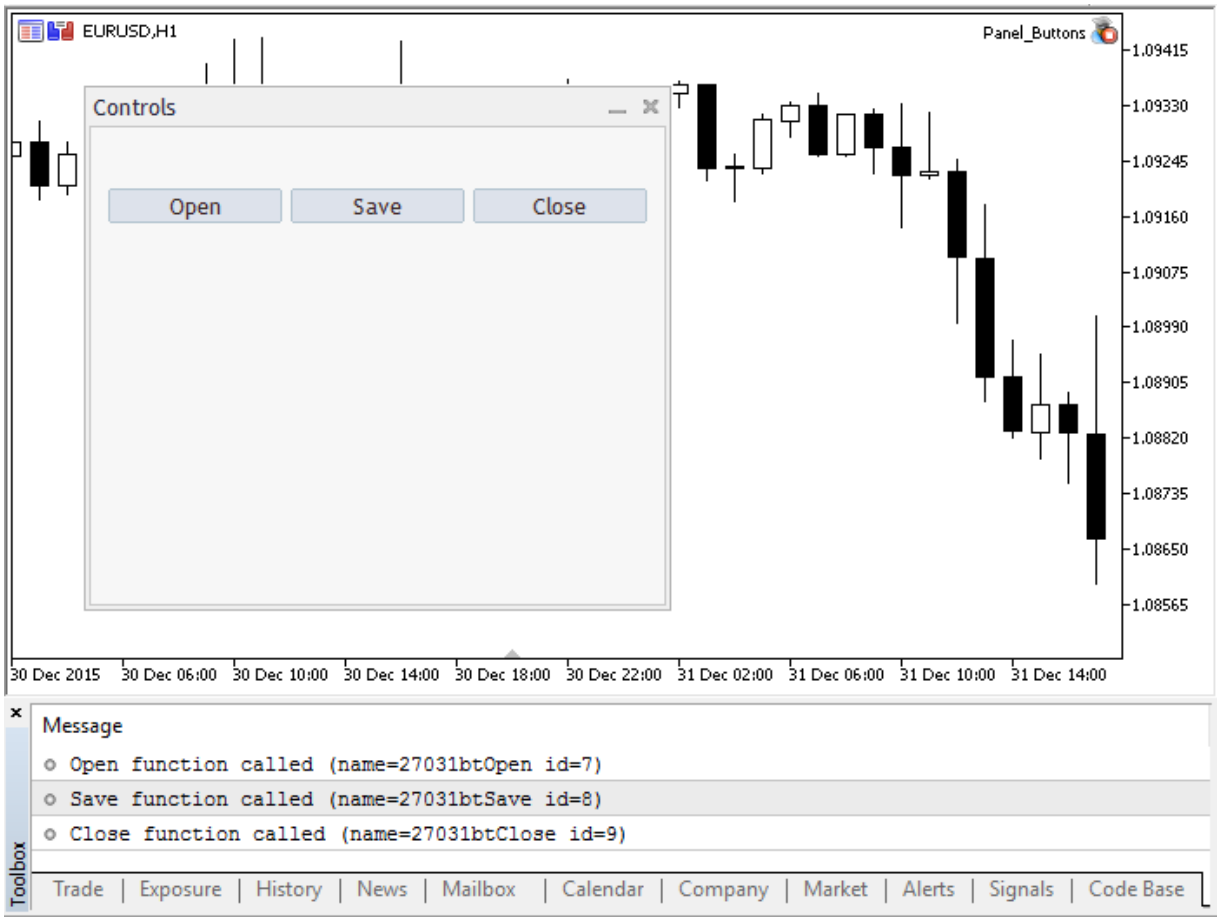
```
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Criação e adição de botões para o painel CControlsDialog          |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- cálculo de coordenadas de botões
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- adicionamos os objetos dos botões juntamente com os ponteiros para as funções
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- criamos os botões graficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- adicionamos cada botão no recipiente CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
```

Agora podemos escrever o programa usando o painel de controle CControlsDialog, no qual são criados 3 botões "Open", "Save" e "Close". Ao pressionar o botão, é chamada a função correspondente que está escrita como um ponteiro para a função *TAction*.

```
//--- declaramos o objeto no nível global para criá-lo automaticamente ao inciar o pro
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- agora criamos o objeto no gráfico
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- executamos o aplicativo
   MyDialog.Run();
//--- inicialização bem-sucedida do aplicativo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy dialog
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
//--- para os eventos do gráfico, chamamos o manipulador a partir da classe mãe (neste
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

A aparência do aplicativo em execução e os resultados dos botões pressionados são mostrados na imagem.

© 2000-2024, MetaQuotes Ltd.



Código-fonte completo do programa

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

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

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

//--- criamos o tipo personalizado de função
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Abre o arquivo                                                  |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Salva o arquivo                                                 |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Fecha o arquivo                                                 |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("Função chamada %s (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Criamos nossa classe de botão com a função de manipulador de eventos |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // manipulador de eventos para o gra
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- construtor com indicação do texto do botão e ponteiro para a função a fim de
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- definição de função que será chamada a partir do manipulador de eventos OnEve
```

```
   void              SetAction(TAction act){m_action=act;}
   //--- manipulador padrão de eventos de gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL & lparam==Id())
        {
         //--- chamamos o manipulador próprio
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- retornamos o resultado da chamada do manipulador a partir da classe mão CE
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| Classe CControlsDialog                                           |
//| Designação: painel gráfico para controle do aplicativo       |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // matriz de botões
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- adição de botão
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- criação de botões
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| Criação do objeto CControlsDialog no gráfico                     |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| Criação e adição de botões para o painel CControlsDialog         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

```
//--- cálculo de coordenadas de botões
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- adicionamos os objetos dos botões juntamente com os ponteiros para as funções
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- criamos os botões graficamente
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("Failed to create button %s %d",b.Text(),i);
         return(false);
        }
      //--- adicionamos cada botão no recipiente CControlsDialog
      if(!Add(b))
         return(false);
     }
//--- succeed
   return(true);
  }
//--- declaramos o objeto no nível global para criá-lo automaticamente ao inciar o pro
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- agora criamos o objeto no gráfico
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- executamos o aplicativo
   MyDialog.Run();
//--- inicialização bem-sucedida do aplicativo
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy dialog
   MyDialog.Destroy(reason);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
//--- para os eventos do gráfico, chamamos o manipulador a partir da classe mãe (neste
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Veja também

Variáveis, Funções

© 2000-2024, MetaQuotes Ltd.

Ponteiros de Objeto

Em MQL5, é possível criar dinamicamente objetos de tipo complexo. Isso é feito usando o operador new, que retorna um identificador para o objeto criado. O descritor tem um tamanho de 8 bytes. Sintaticamente, os descritores de objetos em MQL5 são semelhantes a ponteiros em C++.

Exemplo:

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

Ao contrário do C++, a variável hobject do exemplo acima não é um ponteiro para a memória, senão um descritor para um objeto. Além disso, na linguagem MQL5, todos os objetos nos parâmetros da função devem ser passados por referência. Exemplos de passagem de objetos como parâmetro de função:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- construtores e destruidores
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- inicializamos o objeto Foo
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaramos o objeto como uma variável com criação automática
   Foo foo1;
//--- variante de passagem de objeto por referência
   PrintObject(foo1);

//--- declaramos um ponteiro para um objeto e o criamos usando o operador 'new'
   Foo *foo2=new Foo("foo2");
//--- variante de passagem de um ponteiro para um objeto por referência
   PrintObject(foo2); // o ponteiro para o objeto é convertido automaticamente pelo co

//--- declaramos um array de objetos Foo
```

© 2000-2024, MetaQuotes Ltd.

```
   Foo foo_objects[5];
//--- variante de passagem de array de objetos
   PrintObjectsArray(foo_objects); // função separada para passagem de array de objeto

//--- declaramos um array de ponteiros para objetos do tipo Foo
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- variante de passagem de array de ponteiros
   PrintPointersArray(foo_pointers); // função separada para passagem de array de pont

//--- antes de concluir o trabalho, sempre devemos excluir os objetos criados como por
   delete(foo2);
//--- removemos o array de ponteiros
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//| Os objetos são sempre passados por referência                    |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Passagem de um array de objetos                                  |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| Passagem de array de ponteiros para um objeto                    |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Verificação do ponteiro antes de seu uso

© 2000-2024, MetaQuotes Ltd.

A tentativa de acesso a um ponteiro inválido causa o encerramento crítico do programa. Para verificar o ponteiro antes de usá-lo, usamos a função CheckPointer. O ponteiro pode ser inválido quando:

- o ponteiro é NULL;
- o objeto foi destruído usando o operador delete.

Esta função pode ser usada como verificação de ponteiro. Um valor diferente de zero garante que os dados possam ser acessados com base nesse ponteiro.

```
class CMyObject
 {
protected:
  double              m_value;
public:
                      CMyObject(void);
                      CMyObject(double value) {m_value=value;};
                     ~CMyObject(void){};
  //---
  double              Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- criamos um objeto não inicializado
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

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

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

© 2000-2024, MetaQuotes Ltd.

```
*/
```

Para verificação rápida do ponteiro, você também pode usar o operador "!" (LNOT) que testa sua validade por meio de uma chamada de função implícita CheckPointer. Isso permite que você escreva o código de forma mais concisa e clara. Veja como seriam as verificações do exemplo anterior:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- criamos um objeto não inicializado
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

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

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

Para uma verificação rápida de NULL, use o operador "==". Por exemplo: ptr==NULL ou ptr!=NULL.

Também Veja

Variáveis, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Referência, Modificador & e Palavra-chave this

Passando Parâmetros por Referência

Em MQL5, parâmetros de tipos simples podem ser passados tanto por valor como por referência, enquanto que parâmetros de tipos compostos são sempre passados por referência. Para informar ao compilador que um parâmetro deve ser passado por referência, o caractere "e comercial" **&** é adicionado antes do nome do parâmetro.

Passar um parâmetro por referência significa passar o endereço da variável, é por isso que todas as mudanças no parâmetro que é passado por referência será imediatamente refletida na variável origem. Usando parâmetros passados por referência, pode-se implementar o retorno de vários resultados de uma função ao mesmo tempo. A fim de evitar alteração de um parâmetro passado por referência, use o modificador const.

Assim, se o parâmetro de entrada de uma função for um array, um objeto de classe ou estrutura, o símbolo '&' é colocado no cabeçalho da função após o tipo da variável e antes de seu nome.

Exemplo

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

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

No exemplo acima, a classe CDemoClass é declarada, e contém o membro array private - m_array[] de tipo double. A função setArray() é declarada, e o parâmetro array[] é passado por referência. Se o cabeçalho da função não contém a indicação de passar por referência, isto é, não contém o caractere "e comercial" (&), uma mensagem de erro será gerada na tentativa de compilar tal código.

Apesar do fato de que um array é passado por referência, não se pode atribuir um array ao outro. Nós precisamos realizar a cópia elemento a elemento de conteúdo do array origem para o array destino. A presença de & nas descrições de função é uma condição obrigatória para array e estruturas quando passado como parâmetro de função.

Palavra-chave this

© 2000-2024, MetaQuotes Ltd.

Uma variável de tipo de classe (objeto) pode ser passado tanto por referência como por ponteiro. Assim como referência, o ponteiro permite ter acesso a um objeto. Após o ponteiro de objeto ser declarado, o operador new deve ser aplicado nele para criá-lo e inicializá-lo.

A palavra reservada **this** é destinado para obter a referência do objeto para si mesmo, que fica disponível dentro de métodos de classe e estrutura. **this** sempre referencia o objeto, dentro do método em que ele é usado, e a expressão GetPointer(this) retorna o ponteiro do objeto, cujo membro é a função na qual a chamada ao GetPointer() é realizado. Em funções MQL5 não se pode retornar objetos, mas eles podem retornar ponteiro de objeto.

Assim, se precisamos de uma função para retornar um objeto, nós podemos retornar o ponteiro deste objeto na forma de GetPointer(this). Vamos adicionar a função getDemoClass() que retorna o ponteiro do objeto para este classe, na descrição de CDemoClass.

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

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| Preenchendo o array                                              |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| Retorna seu próprio ponteiro                                     |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Estruturas não têm ponteiros, os operadores *new* e *delete* não podem ser aplicados a eles, GetPointer(this) não pode ser usado.

Também Veja

Ponteiros de Objetos, Criando e Excluindo Objetos, Visibilidade Escopo e Tempo de Vida de Variáveis

© 2000-2024, MetaQuotes Ltd.

Operações e Expressões

Alguns caracteres e seqüências de caractere são de uma importância especial. Existem os chamados símbolos de operação, por exemplo:

```
+ - * / %      Símbolos de operações aritméticas
&& ||          Símbolos de operações lógicas
= += *=        Caracteres de operador de atribuição
```

Os símbolos de operação são usados em expressões e têm sentido quando operandos apropriados são dados a eles. Sinais de pontuação são enfatizados, da mesma forma. Existem parênteses, chaves, vírgula, dois pontos, e ponto e vírgula.

Símbolos de operação, sinais de pontuação, e espaços são usados para separar elementos de linguagem uns dos outros.

Esta seção contém a descrição dos seguintes tópicos:

- Expressões
- Operações Aritméticas
- Operações de Atribuição
- Operações de Relação
- Operações Booleanas
- Operações Binárias
- Outras Operações
- Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Expressões

Uma expressão consiste de um ou mais operandos e símbolos de operação. Uma expressão pode ser escrita em várias linhas.

Exemplos:

```
a++; b = 10;          // várias expressões estão localizados em uma linha
//---  uma expressão está dividida em várias linhas
x = (y * z) /
    (w + 2) + 127;
```

Uma expressão que termina com um ponto e vírgula (;) é um operador.

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Operações Aritméticas

Operações aritméticas incluem operações aditivas e multiplicativas:

```
Soma de variáveis                        i = j + 2;
Diferença de variáveis                   i = j - 3;
Mudança de sinal                         x = - x;
Produto de variáveis                     z = 3 * x;
Quociente de divisão                     i = j / 5;
Resto de divisão                         minutes = time % 60;
Adicionando 1 ao valor da variável       i++;
Adicionando 1 ao valor da variável       ++i;
Subtraindo 1 ao valor da variável        k--;
Subtraindo 1 do valor da variável        --k;
```

Operações de incremento e decremento são aplicadas somente a variáveis, elas não podem ser aplicadas a constantes. O prefixo de incremento (++i) e decremento (--k) são aplicados à variável imediatamente antes desta variável ser usada em uma expressão.

Pós-incremento (i++) e pós-decremento (k--) são aplicados à variável imediatamente depois da variável ser usado em uma expressão.

Observação importante

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

Problemas computacionais podem ocorrer na movimentação da expressão acima de um ambiente de programação para outro (por exemplo, do Borland C++ para MQL5). Geralmente, a ordem de computação depende da implementação do compilador. Na prática, existem duas formas de implementar o pós-decremento (pós-incremento):

1. O pós-decremento (pós-incremento) é aplicado à variável depois de se calcular a expressão toda.
2. O pós-decremento (pós-incremento) é aplicado à variável imediatamente na operação.

Atualmente, a primeira forma de calcular o pós-decremento (pós-incremento) está implementado no MQL5. Mas mesmo conhecendo esta peculiaridade, não se recomenda experiências com seu uso.

Exemplos:

```
int a=3;
a++;            // expressão válida
int b=(a++)*3;  // expressão inválida
```

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Operações de Atribuição

O valor da expressão que inclui uma dada operação é o valor a esquerda do operando após a atribuição:

```
Atribuindo o valor x para a variável y                                  y = x;
```

As seguintes operações unem operações ariméticas e binárias com operações de atribuição:

```
Adicionando x à variável y                                              y += x;
Subtraindo x da variável y                                              y -= x;
Multiplicando a variável y por x                                        y *= x;
Dividindo a variável y por x                                            y /= x;
Resto da divisão da variável y por x                                    y %= x;
Operação AND bit a bit de representações binárias de y e x              y &= x;
Operação AND bit a bit de representações binárias de y e x              y &= x;
Operação AND bit a bit de representações binárias de y e x              y &= x;
Operação OR bit a bit de representações binárias de y e x               y |= x;
Operação OR Exclusivo bit a bit de representações binárias de y e x     y ^= x;
```

Operações bit a bit podem ser aplicados somente em inteiros. Ao realizar a operação de deslocamento lógico da representação de y para direita/esquerda em x bits, os 5 menores dígitos binários do valor de x são usados, os bits maiores são descartados, isto é, o deslocamento é feito para 0-31 bits.

Pela operação %= (resto da divisão de y por x), o sinal resultante é igual ao sinal do número dividido.

O operador de atribuição pode ser usado várias vezes em uma expressão. Neste caso o processamento da expressão é realizado da esquerda para direta:

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

Primeiro, a variável x será atribuída o valor 3, então a variável y será atribuída o valor de x, isto é, também 3.

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Operações de Relação

A booleana FALSE é representada pelo valor zero inteiro, enquanto que a booleana TRUE é representada por qualquer valor não inteiro.

O valor de expressões contendo operações de relação ou operações lógicas é FALSE (0) ou TRUE (1).

```
True se a é igual a b                    a == b;
True se a não igual a b                  a != b;
True se a é menor que b                  a < b;
True se a não é menor que b              a > b;
True se a é menor ou igual a b           a <= b;
True se a é maior ou igual a b           a >= b;
```

A igualdade de dois números reais não pode ser comparada. Na maioria dos casos, dois números aparentemente idênticos podem ser desiguais por causa da diferença de valores na 15 casa decimal. A fim de comparar corretamente dois números reais, compare a diferença normalizada destes números com zero.

Exemplo:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f e %.16f são iguais",first,third);
     }
  }
// Resultado: 0.3000000000000000  0.2999999999999998   são iguais
```

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Operações Booleanas

Negação Lógica NOT (!)

Operando da negação lógia (!) deve ser de tipo aritmético. O resultado é TRUE (1), se o valor do operando é FALSE (0); e é igual a FALSE (0), se o operando difere de FALSE (0).

```
if(!a) Print("não 'a'");
```

Operação lógica OR (||)

Operação OR lógico (||) entre valores x e y. O valor da expressão é TRUE (1), se os valores x ou y são verdadeiros (not null). Senão - FALSE (0).

```
if(x<0 || x>=max_bars) Print("fora da faixa");
```

Operação lógica AND (&&)

Operação lógica AND (&&) entre valores de x e y. O valor da expressão é TRUE (1), se os valores de x e y são verdadeiros (not null). Senão - FALSE (0).

Estimação breve de Operações Booleanas

O esquema do chamado "brief estimate" (estimação breve) é aplicado a operações booleanas, isto é, o cálculo da expressão é finalizado quando o resultado da expressão pode ser precisamente estimado.

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- o primeiro exemplo de estimação breve
   if(func_false() && func_true())
     {
      Print("Operação &&: Você nunca verá esta expressão");
     }
   else
     {
      Print("Operação &&: Resultado da primeira expressão é falso, então a segunda não
     }
//--- o segundo exemplo da estimação breve
   if(!func_false() || !func_true())
     {
      Print("Operação ||: Resultado da primeira expressão é verdadeiro,então a segunda
     }
   else
     {
      Print("Operação ||: Você nunca verá esta expressão");
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| a função sempre retorna false                                    |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Função func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| a função sempre retorna true                                     |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Função func_true()");
   return(true);
  }
```

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Operações Binárias

Complementar de Um

Complementar do valor da variável até um. O valor da expressão contém 1 em todos os dígitos onde o valor da variável contem 0, e 0 em todos os demais dígitos onde a variável contem 1.

```
b = ~n;
```

Exemplo:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// O resultado será:
// a = 97   b = -98
```

Deslocamento para Direita

A representação binária de x é deslocada para direita em y dígitos. Se o valor a ser deslocado é de um tipo sem sinal (unsigned type), o deslocamento lógico para a direita é feito, isto é, os bits do lado esquerdo liberados serão preenchido com zeros.

Se o valor a ser desloca é de um tipo com sinal (signed type), o deslocamento aritmético para direita é feito, isto é, os dígitos do lado esquerdo liberados serão preenchidos com o valor do bit de sinal (sign bit) (se o número é positivo, o valor do bit de sinal é 0, se o número é negativo, o valor do bit de sinal é 1).

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

Exemplo:

```
   char a='a',b='b';
   Print("Antes:  a = ",a, "  b = ",b);
//--- deslocamento para direita
   b=a>>1;
   Print("Após:   a = ",a, "  b = ",b);
// O resultado será:
// Antes:  a = 97   b = 98
// Após:   a = 97   b = 48
```

Deslocamento para Esquerda

A representação binária de x é deslocada para esquerda em y dígitos, os dígitos do lado direito liberados são preenchidos com zero.

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

Exemplo:

```
   char a='a',b='b';
   Print("Antes:  a = ",a, "  b = ",b);
//--- deslocamento para a esquerda
```

© 2000-2024, MetaQuotes Ltd.

```
   b=a<<1;
   Print("Após:   a = ",a, "  b = ",b);
// O resultado será:
// Antes:  a = 97   b = 98
// Após:   a = 97   b = -62
```

Não é recomendável deslocar um número maior ou igual de bits que o comprimento da variável deslocada, porque o resultado de tal operação é indefinido.

Operação AND bit a bit

A operação AND bit a bit em representações binárias de x e y. O valor da expressão contem 1 (TRUE) em todos os dígitos onde ambos x e y contém não-zero, e contem 0 (FALSE) em todos os demais dígitos.

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

Exemplo:

```
   char a='a',b='b';
//--- operação AND
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// O resultado será:
// a = 97   b = 98
// a & b = 96
```

Operação OR bit a bit

A operação OR bit a bit de representações binárias de x e y. O valor da expressão contem 1 em todos os dígitos onde x ou y não contém 0, e contém 0 em todos os demais dígitos.

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

Exemplo:

```
   char a='a',b='b';
//--- operação OR
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// O resultado será:
// a = 97   b = 98
// a | b = 99
```

Operação OR Exclusivo bit a bit

A operação OR exclusivo bit a bit (eXclusive OR) de representações binárias de x e y. O valor da expressão contem 1 em todos os dígitos onde x e y tem valores binários diferentes, e contém 0 em todos os demais dígitos.

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

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
   char a='a', b='b';
//--- operação OR Exclusivo
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// O resultado será:
// a = 97   b = 98
// a ^ b = 3
```

Operações bit a bit são realizadas somente com inteiros.

Também Veja

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Outras Operações

Indexação ([])

Ao endereçar o i-ésimo elemento de um array, o valor da expressão é o valor da variável com o número serial i.

Exemplo:

```
array[i] = 3; // Atribui o valor de 3 ao i-ésimo elemento do array.
```

Somente um inteiro pode ser índice de um array. São permitidos arrays com 4 dimensões ou menos. Cada medida é indexada de 0 ao **tamanho da medida** - 1. Em particular, para um array de uma dimensão consistindo de 50 elementos, a referência ao primeiro elemento se assemelhará a array[0], e a referência ao último elemento será array[49].

Ao endereçar além do array, o subsistema em execução gerará um erro crítico, e o programa será interrompido.

Chamando Função com argumentos x1, x2 ,..., xn

Cada argumento pode representar uma constante, variável, ou expressão do correspondente tipo. Os argumentos passados são separados por vírgula e devem estar dentro dos parênteses, o parêntese de abertura deve seguir o nome da função chamada.

O valor da expressão é o valor retornado pela função. Se o valor retornado é do do tipo void, a chamada da função não pode ser colocada a direita de um operador de atribuição. Repare que a ordem de execução x1, ..., xn é garantida.

Exemplo:

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

Operação Vírgula (,)

Expressões separadas por vírgulas são executadas da esquerda para direita. Todos os efeitos colaterais do cálculo da expressão a esquerda podem aparecer antes da expressão a direita ser calculada. O tipo do resultado e o valor coincidem com aqueles da expressão a direita. A lista de parâmetros a ser passada (seja acima) pode ser considerada como um exemplo.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

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

Operador Ponto (.)

Para o acesso direto a membros públicos de estruturas e classes, a operação ponto é usada. Sintaxe:

```
Nome_da_variável_do_tipo_estrutura.Nome_do_membro
```

Exemplo:

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

Operação de Resolução de Escopo (::)

Cada função em um programa MQL5 tem seu próprio escopo de execução. Por exemplo, a função de sistema Print() é realizada em um escopo global. Funções Importadas são chamadas no escopo do correspondente import. Métodos de funções de classes têm o escopo da correspondente classe. A sintaxe do operador de resolução de escopo é o seguinte:

```
[Scope_name]::Function_name(parameters)
```

Se não existir nenhum nome de escopo, isto é uma orientação explícita para usar o escopo global. Se não existir nenhuma operação de resolução de escopo, a função é procurada no escopo mais próximo. Se não existir nenhuma função no escopo local, a procura é conduzida no escopo global.

A resolução de escopo é também usada para definir funções membros de classe.

```
type Class_name::Function_name(parameters_description)
   {
// corpo da função
   }
```

O uso de várias funções do mesmo nome a partir de contextos de execução diferentes em um programa pode causar ambigüidade. A ordem de prioridade das chamadas da função sem especificação de escopo explícita é a seguinte:

1. Métodos de classe. Se nenhuma função com o nome especificado for definida na classe, mova para o próximo nível.
2. Funções do MQL5. Se o idioma não tiver essa função, vá para o próximo nível.

© 2000-2024, MetaQuotes Ltd.

3. Funções globais definidas pelo usuário. Se nenhuma função com o nome especificado for encontrada, mova para o próximo nível.
4. Funções importadas. Se nenhuma função com o nome especificado for encontrada, o compilador retornará um erro.

Para evitar a ambigüidade de chamadas de função, sempre explicitamente especificar o escopo de função usando a operação de resolução de escopo.

Exemplo:

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

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

© 2000-2024, MetaQuotes Ltd.

Operação de Obtenção do Tamanho do Tipo de Dados ou Tamanho do Objeto de Qualquer Tipo de Dados (sizeof)

Usando a operação sizeof o tamanho de memória correspondente a um identificador ou tipo pode ser definido. A operação sizeof é do seguinte formato:

Exemplo:

```
sizeof(expression)
```

Qualquer identificador, ou nome de tipo entre colchetes pode ser usado como uma expressão. Observe que o nome tipo void não pode ser usado, e o identificar não pode pertencer ao campo de bits, ou ser um nome de função.

Se a expressão for o nome de um array estático (isto é, a primeira dimensão é dada), então o resultado é o tamanho do array inteiro (isto é, o produto do número de elementos e o comprimento do tipo. Se a expressão é o nome de um array dinâmico (a primeira dimensão não é especificada), o resultado será o tamanho do objeto do array dinâmico.

Quando sizeof é aplicado ao nome de um tipo de estrutura ou classe, ou ao identificador do tipo de uma estrutura ou classe, o resultado é tamanho real da estrutura ou classe.

Exemplo:

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

O tamanho é calculado na etapa de compilação.

Veja Também

Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Regras de Precedência

Cada grupo de operações na tabela tem a mesma prioridade. Quanto mais alta a prioridade de operações, mais alta é a posição do grupo na tabela. As regras de precedência determinar o agrupamento de operações e operandos.

Atenção: A precedência de operações na linguagem MQL5 corresponde à prioridade adotada em C++, e difere da prioridade dada na linguagem MQL4.

Operação	Descrição	Ordem de Execução
() [] .	Chamada de Função Referenciando um elemento de array Referenciando um elemento de estrutura	Da esquerda para direita
! ~ - ++ -- (tipo) sizeof	Negação lógica Negação bit a bit (complemento) Mudança de Sinal Incremento por um Decremento por um Conversão de Tipo (Typecasting) Determinação de tamanho em bytes	Direita para esquerda
* / %	Multiplicação Divisão Quociente de divisão	Da esquerda para direita
+ -	Adição Subtração	Da esquerda para direita
<< >>	Deslocamento para esquerda Deslocamento para direita	Da esquerda para direita
< <= > >=	Menor que Menor ou igual a Maior que Maior ou igual a	Da esquerda para direita
== !=	Igual Não igual	Da esquerda para direita
&	Operação AND bit a bit	Da esquerda para direita
^	OR exclusivo bit a bit	Da esquerda para direita
|	Operação OR bit a bit	Da esquerda para direita
&&	Operação lógica AND	Da esquerda para direita
||	Operação lógica OR	Da esquerda para direita
?:	Operador condicional	Direita para esquerda
= *=	Atribuição Multiplicação com atribuição	Direita para esquerda

© 2000-2024, MetaQuotes Ltd.

Operação	Descrição	Ordem de Execução
/= %= += -= <<= >>= &= ^= |=	Divisão com atribuição Quociente de divisão com atribuição Adição com atribuição Subtração com atribuição Deslocamento para esquerda com atribuição Deslocamento para direita com atribuição AND bit a bit com atribuição OR Exclusivo com atribuição OR bit a bit com atribuição	
,	Vírgula	Da esquerda para direita

Para mudar a ordem de execução de operação, são usados parênteses, que são de prioridade mais alta.

© 2000-2024, MetaQuotes Ltd.

Operadores

Operadores de linguagem descrevem algumas operações algorítmicas que devem ser executadas para se realizar uma tarefa. O corpo do programa é uma seqüência de tais operadores. Operadores, um a um, são separados por ponto e vírgula.

Operador	Descrição
Operador Composto {}	Um ou mais operadores de qualquer tipo, entre chaves {}
Operador de Expressão (;)	Qualquer expressão que termina com ponto e vírgula (;)
return operador	Finaliza a função corrente e retorna o controle para o programa chamador
if-else operador condicional	É usado quando é necessário fazer uma escolha
?: operador condicional	Um análogo simples do operador condicional if-else
switch operador de seleção	Passa o controle para o operador que corresponde ao valor da expressão
while operador de loop	Realiza uma operação até que a expressão verificada se torne falsa. A expressão é verificada antes de cada iteração
for operador de loop	Realiza uma operação até que a expressão verificada se torne falsa. A expressão é verificada antes de cada iteração
do-while operador de loop	Realiza uma operação até que a expressão verificada se torne falsa. A condição de finalização é verificada, após cada loop. O corpo do loop é executado pelo menos uma vez.
break operador	Finaliza a execução do operador externo adjunto mais próximo: switch, while, do-while ou for.
continue operador	Passa controle para o começo do loop do operador externo mais próximo: while, do-while ou for
new operador	Cria um objeto de tamanho apropriado e retorna um descritor do objeto criado.
delete operador	Exclui o objeto criado pelo operador new

Um operador pode ocupar uma ou mais linhas. Dois ou mais operadores podem ser colocados na mesma linha. Operadores que controlam a ordem de execução (if, if-else, switch, while e for) podem ser aninhados (encaixados) um no outro.

Exemplo:

```
if(Month() == 12)
  if(Day() == 31) Print("Feliz Ano Novo!");
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador Composto

Um operador composto (um bloco) consiste de um ou mais operadores de qualquer tipo, entre chaves {}. A chave de fechamento deve ser seguida por dois pontos (;).

Exemplo:

```
if(x==0)
  {
   Print("posição inválida x = ",x);
   return;
  }
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador de Expressão

Qualquer expressão seguida por ponto e vírgula (;) é o operador. Aqui estão alguns exemplos de operadores de expressão.

Operador de Atribuição

Identificador = expressão;

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

Um operador de atribuição pode ser usado muitas vezes em uma expressão. Neste caso, o processamento da expressão é realizado da direita para a esquerda.

Operador de Chamada de Função

Function_name (argument1,..., argumentN);

```
FileClose(file);
```

Operador Empty

Consiste somente de um ponto e vírgula (;) e é usado para denotar um corpo vazio de um operador de controle.

Veja Também

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador return

O operador return finaliza a execução da função corrente e retorna o controle para o programa chamador. O resultado do cálculo da expressão é retornado para a função chamadora. A expressão pode conter um operador de atribuição.

Exemplo:

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

Em funções com o tipo de retorno void, o operador return sem expressão deve ser usada:

```
void SomeFunction()
  {
   Print("Olá!");
   return;    // este operador pode ser removido
  }
```

A chave direita da função ("}") significa execução implícita do operador return sem expressão.

Pode ser retornado: tipos simples, estruturas simples, ponteiros de objetos. Com o operador *return* não se pode retornar arrays, objetos de classe, variáveis de tipo estrutura composta.

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador Condicional If-Else

O operador IF - ELSE é usado quando uma escolha deve ser feita. Formalmente, a sintaxe é a seguinte:

```
if (expressão)
    operador1
else
    operador2
```

Se a expressão é verdadeira, o operador1 é executado e o controle é dado ao operador que segue o operador2. Se a expressão é falsa, o operador2 é executado.

A parte else do operador if pode ser omitida. Assim, uma divergência pode aparecer em operadores aninhados (encaixados) if com omissão da parte else. Neste caso, o else se refere ao operador anterior if mais próximo no mesmo bloco que não tem a parte else.

Exemplos:

```
//--- A parte else refere-se ao segundo operador if:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- A parte else refere-se ao primeiro operador if:
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- Operadores aninhados
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador Ternário ?

A forma geral do operador ternário é a seguinte:

```
expressão1 ? expressão2 : expressão3
```

Para o primeiro operando - "expressão1" - qualquer expressão que resulte em um valor do tipo bool pode ser usado. Se o resultado é true, então o operador definido pelo segundo operando, isto é, "expressão2", é executado.

Se o primeiro operando é false, o terceiro operando - "expressão3" é executado. Os segundo e terceiro operandos, isto é, "expressão2" e "expressão3" devem retornar valores de algum tipo e não devem ser do tipo void. O resultado da execução do operador condicional é o resultado de expressão2 ou o resultado de expressão3, dependendo do resultado de expressão1.

```
//--- diferença normalizada entre preços de abertura e fechamento para intervalo de ur
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Esta entrada é equivalente ao seguinte:

```
   double true_range;
   if(High==Low)true_range=0;                // se o Máximo e o Mínimo são iguais
   else true_range=(Close-Open)/(High-Low); // se a faixa não é nula
```

Restrições ao Uso do Operador

Baseado no valor da "expressão1", o operador deve retornar um dentre dois valores - ou "expressão2" ou "expressão3". Existem várias limitações nessas expressões:

1. Não confunda o tipo user-defined (definida pelo usuário) com o tipo simples ou enumerador. NULL pode ser usado para ponteiro.
2. Se os tipos dos valores são simples, o operador será do tipo máximo (veja Conversão de Tipos (type casting)).
3. Se algum dos valores é uma enumeração e o segundo valor é de um tipo numérico, a enumeração é substituída por int e a segunda regra é aplicada.
4. Se ambos os valores são enumerações, seus tipos devem ser idênticos, e o operador será do tipo enumeração.

Restrições ao tipos user-defined (classes ou estruturas definidas pelo usuário):

a) Os tipos devem ser idênticos ou um deve ser derivado do outro.

b) Se os tipo não forem idênticos (herança), então a filha é implicitamente convertida para o pai, isto é, o operador será do tipo paterno.

c) Não confunda objetos com ponteiros - ambas as expressão ou são objetos ou são ponteiros. NULL pode ser usado para ponteiro.

Observação

© 2000-2024, MetaQuotes Ltd.

Cuidado ao usar o operador condicional como um argumento de uma função de sobrecarga, porque o tipo do resultado de um operador condicional é definido na durante a compilação do programa. E este tipo é definido como a maior dos tipos "expression2" e "expression3".

Exemplo:

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

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

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

   func(Expression1?Expression2:Expression3);   // aviso sobre conversão implícita pa
   func(!Expression1?Expression2:Expression3);  // aviso sobre conversão implícita pa
  }

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

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador switch

Compara o valor da expressão com constantes em todas as variantes de *case* e passa o controle para o operador, que corresponde ao valor da expressão. Cada variante de *case* pode ser marcada com uma constante inteira, uma constante literal ou uma expressão constante. A expressão constante não pode conter variáveis ou chamadas de função. A expressão do operador *switch* deve ser do tipo inteiro - int ou uint.

```
switch(expressão)
  {
   case constantes: operadores
   case constantes: operadores
      ...
   default: operadores
  }
```

Operadores marcados pelo rótulo *default* são executados se nenhuma das constantes nos operadores *case* forem iguais ao valor da expressão. A variante *default* não precisa ser necessariamente declarada e não precisa ser necessariamente ser a último. Se nenhuma das constantes corresponder ao valor da expressão e a variante *default* não estiver presente, nenhuma ação será executada.

A palavra-chave *case* com uma constante são apenas rótulos, e se operadores forem executados para alguma variante *case*, o programa continuará a executar os operadores de todas as subseqüentes variantes até que o operador *break* ocorra. Isso permite vincular uma seqüência de operadores com várias variantes.

Uma expressão constante é calculada durante a compilação. Duas constantes em um operador *switch* não podem ter o mesmo valor.

Exemplos:

```
//--- Primeiro exemplo
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B ou C");
      break;
   default:
      Print("NOT A, B ou C");
      break;
  }

//---  Segundo exemplo
   string res="";
   int i=0;
   switch(i)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      case 1:
         res=i;break;
      default:
         res="default";break;
      case 2:
         res=i;break;
      case 3:
         res=i;break;
     }
   Print(res);
/*
   Result
   default
*/
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador de Loop while

O operador while consiste de uma expressão verificada e um operador:

```
while(expression)
 operador;
```

Se a expressão é verdadeira, o operador é executado até que a expressão se torne falsa. Se a expressão é falsa, o controle é passado para o próximo operador. O valor da expressão é definido antes do operador ser executado. Portanto se a expressão é falsa logo no começo, o operador não será executado.

Observação

Se é esperado que um grande número de iterações seja tratado por um loop, é recomendável que se verifique a ocorrência de uma finalização de programa forçada usando a função IsStopped().

Exemplo:

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

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador de Loop For

O operador for consiste de três expressões e um operador executável:

```
for(expressão1; expressão2; expressão3)
  operador;
```

Expressão1 descreve a inicialização do loop. Expression2 verifica as condições de finalização do loop. Se for verdadeira, o corpo do loop do for é executado. Realiza uma operação até que a expressão verificada se torne falsa. Se a expressão é verdadeira, o operador1 é executado e o controle é dado ao operador que segue o operador2. Expression3 é calculado após cada iteração.

O operador for é equivalente à seguinte sucessão de operadores:

```
expressão1;
while(expressão2)
  {
  operador;
  expressão3;
  };
```

Quaisquer das três ou todas as três expressões podem estar ausentes no operador for, mas o ponto e vírgula (;) que as separada não pode ser omitido. Se expressão2 é omitida, ela é considerada como verdadeira. O operador for(;;) é um loop contínuo, equivalente ao operador while(1). As expressões 1 e 3 podem consistir de várias expressões combinadas por um operador vírgula ','.

Observação

Se é esperado que um grande número de iterações seja tratado por um loop, é recomendável que se verifique a ocorrência de uma finalização de programa forçada usando a função IsStopped().

Exemplos:

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

Também Veja

© 2000-2024, MetaQuotes Ltd.

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador de loop do while

Os loops for e while verificam o término no começo, não no fim de um loop. O terceiro operador de loop do - while verifica a condição de término no fim, após cada interação do loop. O corpo do loop é sempre executado pelo menos uma vez.

```
do
  operador;
while(expression);
```

Primeiro o operador é executado, então a expressão é calculada. Se a expressão for verdadeira, então o operador é executado novamente, e assim por diante. Se a expressão se tornar falsa, o loop termina.

Observação

Se é esperado que um grande número de iterações seja tratado por um loop, é recomendável que se verifique a ocorrência de uma finalização de programa forçada usando a função IsStopped().

Exemplo:

```
//--- Calcula a série de Fibonacci
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // Sem este operador um loop infinito aparecerá!
     }
   while(i<counterFibonacci && !IsStopped());
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador break

O operador break finaliza a execução do operador aninhado (encaixado) switch, while, do-while ou for mais próximo. O controle é passado ao operador que se segue a aquele finalizado. Um dos propósitos deste operador é finalizar a execução de loop quando um certo valor é atribuído a uma variável.

Exemplo:

```
//--- buscando pelo primeiro elemento zero
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador continue

O operador continue passa o controle para o começo do próximo loop do operador while, do-while ou for mais próximo, executando a próxima iteração. O propósito deste operador é oposto ao do operador break.

Exemplo:

```
//--- Soma de todos os elementos não-zero
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
     }
   return(sum);
  }
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operador de Criação de Objeto new

O operador new automaticamente cria um objeto de tamanho correspondente, chama o construtor do objeto e retorna um descritor do objeto criado. Em caso de falha, o operador retorna um descritor de null que pode ser comparado com a constante NULL.

O operador new pode ser aplicado somente a objetos de classe. Ele não pode ser aplicado a estruturas.

O operador não deve ser usado para criar arrays de objetos. Para fazer isso, use a função ArrayResize().

Exemplo:

```
//+------------------------------------------------------------------+
//| Criação de Figura                                                |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- cria aleatoriamente uma das 7 possíveis formas
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- desenhar
   if(m_shape!=NULL)
     {
      //--- pré-configurações
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- draw
      m_shape.Draw();
     }
//---
  }
```

Deve ser notado que um descritor do objeto não é um ponteiro para um endereço de memória.

Um objeto criado com o operador new deve ser explicitamente removido usando o operador delete.

Também Veja

© 2000-2024, MetaQuotes Ltd.

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Operação de Exclusão de Objeto delete

O operador delete exclui um objeto criado pelo operador new, chama o destrutor da classe correspondente e libera memória ocupada pelo objeto. Um descritor real de um objeto existente é usado como um operando. Após a operação de exclusão (delete) ser executada, o descritor do objeto torna-se inválido.

Exemplo:

```
//--- exclui figura
delete m_shape;
m_shape=NULL;
//--- criar uma nova figura
NewShape();
```

Também Veja

Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Funções

Toda tarefa pode ser dividida em sub-tarefas, cada qual podendo ser representada tanto na forma de código, como ser divida em sub-tarefas ainda menores. Este método é chamado de *refinamento passo a passo.*. Funções são usadas para escrever o código das sub-tarefas a serem resolvidas. O código que descreve o que uma função faz é chamado de *definição de função:*

```
function_header
  {
  instruções
  }
```

Tudo que está antes da primeira chave é o *cabeçalho (header)* da definição de função, e o que está entre as chaves é o *corpo (body)* da definição de função. O cabeçalho de função (function header) inclui a descrição do tipo de valor de retorno, o nome (identificador) e os parâmetros formais. O número de parâmetros passados para a função é limitado e não pode exceder 64.

A função pode ser chamada de outras partes do programa quantas vezes forem necessárias. Na verdade, o tipo de retorno, o identificador da função e os tipos de parâmetros constituem o *protótipo da função*.

Protótipo de função é a declaração da função, mas não a sua definição. Devido a declaração explícita do tipo de retorno e da lista dos tipos de argumentos, a verificação estrita de tipo e a implícita conversão de tipo (typecasting) são possíveis durante as chamadas de função. Muito freqüentemente declarações de funções são usadas em classes para melhorar a legibilidade do código.

As definições de função devem corresponder exatamente à sua declaração. Cada função declarada deve ser definida.

Exemplo:

```
double                        // tipo do valor de retorno
linfunc (double a, double b)  // nome da função e lista de parâmetros
  {
                              // operador composto
   return (a + b);            // valor do retorno
  }
```

O operador return pode retornar o valor de uma expressão localizada neste operador. Se necessário, o valor da expressão é convertido para o tipo do resultado da função. Pode ser retornado: tipos simples, estruturas simples, ponteiros de objetos. Com o operador *return* não se pode retornar arrays, objetos de classe, variáveis de tipo estrutura composta.

Uma função que não retorna nenhum valor deve ser descrita como do tipo void.

Exemplo:

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

© 2000-2024, MetaQuotes Ltd.

Parâmetros passado para a função podem ter valores default (padrão), que são definidos por constantes daquele tipo.

Exemplo:

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="string transmitida")
  {
   Print("Parâmetro obrigatório a = ",a);
   Print("Transmitir os seguintes parâmetros: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

Se algum dos parâmetros tem um valor default, todos os parâmetros subseqüentes devem também ter valores default.

Exemplo de declaração incorreta:

```
int somefunc(double a,
             double d=0.0001,     // valor padrão declarado 0.0001
             int n,               // não é determinado o valor padrão!
             bool b,              // não é determinado o valor padrão!
             string s="string transmitida")
  {
  }
```

Também Veja

Sobrecarga (Overload), Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Chamada de Função

Se um nome, que não foi descrito antes, aparece em uma expressão e é seguido pelo parêntesis de abertura, ele é considerado contextualmente como o nome de uma função.

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

Argumentos (parâmetros formais) são passados por valor, isto é, cada expressão x1,.., xn é calculada, e os valores são passados para a função. A ordem de cálculo das expressões e a ordem de carregamento dos valores não são garantidos. Durante a execução, o sistema verifica o número e o tipo dos argumentos passados para a função. Tal forma de abordar uma função é chamada de uma chamada de valor.

Chamada de função é uma expressão, cujo valor é o valor retornado pela função. O tipo da função descrito acima deve corresponder com o tipo do valor de retorno. Uma função pode ser declarada ou descrita em qualquer parte do programa no escopo global, isto é, fora de outras funções. Uma função não pode ser declarada ou descrita dentro de outra função.

Exemplos:

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

Na chamada de uma função com parâmetros default, a lista de parâmetros a serem passados pode ser limitada, mas não antes do primeiro parâmetro default.

Exemplos:

```
void somefunc(double init,
              double sec=0.0001, //define valores default
              int level=10);
//...
somefunc();                      // Chamada errada. O primeiro parâmetro deve estar p
somefunc(3.14);                  // Chamada correta
somefunc(3.14,0.0002);           // Chamada correta
somefunc(3.14,0.0002,10);        // Chamada correta
```

Ao chamar uma função, não se pode pular parâmetros, mesmo aqueles que têm valores default:

```
somefunc(3.14, , 10);           // Chamada errada -> o segundo parâmetros foi pulado.
```

Também Veja

Sobrecarga (Overload), Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Passando Parâmetros

Existem pois métodos pelo qual a linguagem de máquina pode passar argumentos para um subprograma (função). O primeiro método é enviar um parâmetro por valor. Este método copia o valor do argumento para um parâmetro formal de função. Portanto, quaisquer mudanças neste parâmetro ocorridas dentro da função não têm influência no correspondente argumento usado na chamada da função.

```
//+------------------------------------------------------------------+
//| Passando parâmetros por valor                                    |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a e b antes chamada:",a," ",b);
   double d=FirstMethod(a,b);
   Print("a e b após chamada:",a," ",b);
  }
//--- Resultado da execução do script
//  a e b antes chamada: 14 8
//  a e b após chamada: 14 8
```

O segundo método é passar por referência. Neste caso, a referência para um parâmetro (não seu valor) é passada para um parâmetro de função. Dentro da função, ele é usado para referenciar o verdadeiro parâmetro especificado na chamada. Isso significa que mudanças no parâmetro afetarão o argumento usado na chamada da função.

```
//+------------------------------------------------------------------+
//| Passando parâmetros por referência                               |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
```

© 2000-2024, MetaQuotes Ltd.

```
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("a e b antes chamada:",a," ",b);
   double d=SecondMethod(a,b);
   Print("a e b após chamada:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- resultado da execução do script
//  a e b antes chamada: 14 8
//  a e b após chamada: 28 4
```

MQL5 usa ambos os métodos, com uma exceção, arrays, variáveis tipo estrutura e objetos de classe são sempre passados por referência. A fim de evitar modificações nos parâmetros reais (argumentos passados na chamada da função) use o especificador de acesso const. Ao se tentar modificar o conteúdo de uma variável declarada com o especificador *const*, o compilador gerará um erro.

Observação

Deve se notar que os parâmetros são passados para uma função em ordem inversa, ou seja, o último parâmetro é calculado e passado primeiro, depois o último mas apenas um, etc. O último parâmetro calculado e passado é aquele que está em primeiro lugar depois da abertura dos parênteses.

Exemplo:

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

   func(a[i],a[i++],"First call (i = "+string(i)+")");
   func(a[i++],a[i],"Second call (i = "+string(i)+")");
// Result:
// First call (i = 0) : par1 = 1     par2 = 0
// Second call (i = 1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void func(int par1,int par2,string comment)
```

© 2000-2024, MetaQuotes Ltd.

```
{
 Print(comment,": par1 = ",par1,"    par2 = ",par2);
}
```

Na primeira chamada (ver exemplo acima) a variável ***i*** é usada pela primeira vez na concatenação de strings:

```
"Primeira chamada (i = "+string(i)+")"
```

Aqui o valor não muda. Então a variável ***i*** é usada no cálculo do elemento do array ***a[i++]***, ou seja, quando o elemento do array com índice i é acessado, a variável ***i*** é incrementada. E só depois disso o primeiro parâmetro com valor alterado da variável ***i*** é calculado.

Na segunda chamada o mesmo valor de i (calculado sobre a primeira fase da função chamada) é utilizado no cálculo de todos os três parâmetros. Somente depois de os primeiros parâmetros serem calculados, a variável ***i*** é alterada novamente.

Também Veja

Escopo de visibilidade e tempo de vida das variáveis, Sobrecarga, Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Sobrecarga de Função

Geralmente, o nome da função tende a refletir seu propósito principal. Como regra, programas legíveis contém vários identificadores bem selecionados. Às vezes, funções diferentes são usados para os mesmos propósitos. Vamos considerar, por exemplo, uma função que calcula o valor médio de um array de números de dupla precisão e a mesma função, mas operando com um array de inteiros. Ambas são convenientemente chamadas de AverageFromArray:

```
//+------------------------------------------------------------------+
//| O cálculo de média de um array de tipo double                    |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // Soma para o tipo double
     }
   aver=sum/size;        // Apenas divide a soma pelo número
//---
   Print("Cálculo da média para um array do tipo double");
   return aver;
  }
//+------------------------------------------------------------------+
//| O cálculo de média para um array de tipo int                     |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];      // Soma para o tipo int
     }
   aver=(double)sum/size;// Fornece uma quantidade de tipo double, e divide
//---
   Print("Cálculo da média para um array do tipo int");
   return aver;
  }
```

Cada função contém a saída da mensagem via a função Print();

```
   Print("Cálculo da média para um array do tipo int");
```

© 2000-2024, MetaQuotes Ltd.

O compilador seleciona a função necessária de acordo com os tipos dos argumentos e suas quantidades. A regra, que define a escolha a ser feita, é chamada de *algoritmo de correspondência de assinatura*. Um assinatura é uma lista de tipos usados na declaração da função.

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- Resultado do script
// Calcula a média para um array de tipo int
// Calcula a média para um array de tipo double
// int_aver= 3.00000000    double_aver= 3.30000000
```

Sobrecarga de função é um processo de criar várias funções com o mesmo nome, mas com diferentes parâmetros. Isso significa que em variantes sobrecarregadas de uma função, o número de argumentos e/ou seus tipos devem ser diferentes. Um variante de função específica é selecionada baseada na correspondência entre lista de argumentos ao chamar a função com a lista de parâmetros na declaração da função.

Quando uma função sobrecarregada é chamada, o compilador deve ter uma algoritmo para selecionar a função apropriada. O algoritmo que realiza esta escolha depende da conversão de quais tipos estiverem presentes. A melhor correspondência deve ser única. Uma função sobrecarregada deve ser a melhor correspondência dentre todas as outras variantes para ao menos um argumento. Ao mesmo tempo, ela não deve ser pior que as outras variantes para todos os outros argumentos.

Abaixo está um algoritmo de correspondência para cada argumento.

Algoritmo de Escolha de uma Função Sobrecarregada

1. Use estrita correspondência (se possível).
2. Tente incremento de tipo padrão.
3. Tente conversão de tipo padrão.

O incremento de tipo padrão é melhor que outras conversões padrão. Incremento é a conversão de float para double, de bool, char, short ou enum para int. Conversão de tipo de arrays de tipos inteiros similares também pertencem a conversão de tipo. Tipos similares são: bool, char, uchar, já que todos os três tipo são inteiros de único bytes, inteiros de duplo byte short e ushort; inteiros de 4 bytes int, uint, e color; long, ulong e datetime.

Claro que a estrita correspondência é a melhor. Para alcançar tal consistência conversão de tipo (typecasting) pode ser usada. O compilador não pode lidar com situações ambíguas. Portanto você não

© 2000-2024, MetaQuotes Ltd.

deve confiar em diferenças sutis de tipos e conversões implícitas que tornam a função sobrecarregada não clara.

No caso de dúvida, use conversão explícita para assegurar estrita adequação.

Exemplos de funções sobrecarregadas no MQL5 podem ser vistas no exemplo de funções ArrayInitialize().

Regras de sobrecarga de função para sobrecarga de métodos de classe.

A sobrecarga de funções de sistema é permitida, mas deve-se observar que o compilador é capaz de selecionar com precisão a função necessária. Por exemplo, podemos sobrecarregar o sistema da função MathMax() de 4 maneiras diferentes, mas apenas duas variantes são corretas.

Exemplo:

```
// 1. sobrecarga é permitida - função difere da função embutida MathMax() em número de
double MathMax(double a,double b,double c);

// 2. sobrecarga não é permitida!
// número de parâmetros é diferente, mas o último tem um valor default
// isso leva ao ocultamento da função de sistema ao chamar-lo, o que é inaceitável
double MathMax(double a,double b,double c=DBL_MIN);

// 3. sobrecarga é permitida - sobrecarga normal para tipos de parâmetros 'a' e 'b'
double MathMax(int a,int b);

// 4. sobrecarga não é permitida!
// o número e tipos de parâmetros são os mesmos que no original double MathMax (double
int MathMax(double a,double b);
```

Também Veja

Sobrecarga (Overload), Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Sobrecarga de Operação

Para facilitar a leitura e escrita de código, a sobrecarga de algumas operações é permitida. O operador de sobrecarga é escrito usando a palavra-chave operator. Os seguintes operadores podem ser sobrecarregados:

- binário +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^
- unário +,-,++,--,!,~
- operador de atribuição =
- operador de indexação []

Sobrecarga de operação permite o uso da notação de operação (escrita na forma de expressões simples) para objetos complexos - estruturas e classes. Escrevendo expressões usando operações de sobrecarga simplifica a visualização do código fonte, porque uma implementação mais complexa fica escondida.

Por exemplo, considere números complexos, que consistem de partes real e imaginária. Eles são amplamente utilizados na matemática. A linguagem MQL5 não tem um tipo de dado que represente números complexos, mas é possível criar um novo tipo de dado na forma de uma estrutura ou classe. Declare a estrutura complexa e defina quatro métodos que implementam as quatro operações aritméticas:

```
//+------------------------------------------------------------------+
//| Uma estrutura para operações com números complexos               |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Parte real
   double            im; // Parte imaginário
   //--- Construtores
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Operações Aritméticas
   complex           Add(const complex &l,const complex &r) const;  // Adição
   complex           Sub(const complex &l,const complex &r) const;  // Subtração
   complex           Mul(const complex &l,const complex &r) const;  // Multiplicação
   complex           Div(const complex &l,const complex &r) const;  // Divisão
  };
```

Agora, em nosso código nós podemos declarar variáveis representando números complexos, e trabalhar com eles.

Por exemplo:

```
void OnStart()
  {
//--- Declara e inicialize variáveis de um tipo complexo
   complex a(2,4),b(-4,-2);
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Soma dois números
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplica dois números
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Dividir dois números
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Mas seria mais conveniente usar os operadores usuais "+", "-", "*" e "/" para operações aritméticas comuns com números complexos.

A palavra-chave operator é usado para definir uma função membro que realiza conversão de tipo. Operações unárias e binárias para variáveis de objeto de classe podem ser sobrecarregadas como funções membro não estáticas. Elas implicitamente agem nos objetos de classe.

A maioria das operações binárias podem ser sobrecarregadas como funções regulares que tomam uma variável de classe e/ou um ponteiro de objeto desta classe como argumento. Para o nosso tipo complexo, a sobrecarga na declaração se parecerá como:

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

O exemplo completo do script:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Declara e inicialize variáveis de um tipo complexo
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- Soma dois números
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplica dois números

   z=a*b;
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Dividir dois números
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| Uma estrutura para operações com números complexos               |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Parte real
   double            im; // Parte imaginário
   //--- Construtores
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- Operações Aritméticas
   complex           Add(const complex &l,const complex &r) const;  // Adição
   complex           Sub(const complex &l,const complex &r) const;  // Subtração
   complex           Mul(const complex &l,const complex &r) const;  // Multiplicação
   complex           Div(const complex &l,const complex &r) const;  // Divisão
   //--- Operadores binárias
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//| Adição                                                           |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- Resultado
   return res;
  }
//+------------------------------------------------------------------+
//| Subtração                                                        |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
```

© 2000-2024, MetaQuotes Ltd.

```
   res.im=l.im-r.im;
//--- Resultado
   return res;
  }
//+------------------------------------------------------------------+
//| Multiplicação                                                    |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- Resultado
   return res;
  }
//+------------------------------------------------------------------+
//| Divisão                                                          |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- Numero complexo vazio
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- Verificar se é zero
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": número é zero");
      return(res);
     }
//--- Variáveis auxiliares
   double e;
   double f;
//--- Selecionando a variante de cálculo
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- Resultado
   return res;
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

A maioria das operações unárias para classes podem ser sobrecarregadas como funções comuns que aceitam um único argumento de objeto de classe ou ponteiro dele. Adicione sobrecarga de operações unárias "-" e "!".

```
//+------------------------------------------------------------------+
//| Uma estrutura para operações com números complexos               |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // Parte real
   double            im;       // Parte imaginário
...
   //--- Operadores unários
   complex operator-()  const; // Unary minus
   bool    operator!()  const; // Negação
  };
...
//+------------------------------------------------------------------+
//| Sobrecarregar operador de "menos unário"                         |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- Resultado
   return res;
  }
//+------------------------------------------------------------------+
//| Sobrecarregar operador de "negação lógica"                       |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- São as partes real e imaginária do número complexo igual a zero?
   return (re!=0 && im!=0);
  }
```

Agora nós podemos verificar se valor de um número complexo é zero e obter um valor negativo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Declara e inicialize variáveis de um tipo complexo
```

© 2000-2024, MetaQuotes Ltd.

```
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Dividir dois números
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- Um número complexo é igual a zero por padrão (no construtor padrão re==0 e im==0
   complex zero;
   Print("!zero=",!zero);
//--- Atribuir um valor negativo
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- Verificar se é zero mais uma vez
   Print("!zero=",!zero);
//---
  }
```

Note que nós não tivemos que sobrecarregar o operador de atribuição "=", já que estruturas de tipos simples pode ser diretamente copiadas uma no outra. Assim, nós agora podemos escrever um código para cálculos envolvendo números complexos de maneira usual.

Sobrecarga de operador de indexação permite obter os valores dos arrays fechados em um objeto, de uma maneira simples e familiar, e isso também contribui para uma melhor legibilidade do código fonte. Por exemplo, nós precisamos fornecer acesso a um símbolo dentro de uma string em uma posição específica. Uma string em MQL5 é um tipo string separado, que não é um array de símbolos, mas com a ajuda de uma operação de indexação sobrecarregada podemos fornecer um trabalho simples e transparente na classe CString gerada:

```
//+------------------------------------------------------------------+
//| Uma classe para acessar símbolos em string como na array de símbolos |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Um array para receber os símbolos a partir de uma string
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                 17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- Fazer um frase usando símbolos da variável str
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- Mostrar o resultado
   Print(res);
  }
```

Um outro exemplo de sobrecarga do operador de indexação são operações com matrizes. A matriz representa um array dinâmico de duas dimensões, o tamanho do array não é definido com antecedência. Portanto, você não pode declarar um array da forma array[][] sem especificar o tamanho da segunda dimensão, e então passar este array como um parâmetro. Uma possível solução é uma classe especial CMatrix, que contém um array de objetos de classe CRow.

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Operações de adição e multiplicação de matrizes
   CMatrix A(3),B(3),C();
//--- Preparar um array para as linhas
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- Preencher as matrizes
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- Saída de matrizes no log Experts
   Print("---- Os elementos da matriz A");
   Print(A.String());
   Print("---- Os elementos da matriz B");
   Print(B.String());

//--- Adição de matrizes
   Print("---- Adição das matrizes A e B");
   C=A+B;
//--- Saída da representação da string formatada
   Print(C.String());

//--- Multiplicação de matrizes
   Print("---- Multiplicação das matrizes A e B");
   C=A*B;
   Print(C.String());

//--- Agora mostraremos como obter valores no estilo dos arrays dinâmicos matrix[i][j]
   Print("Saída de valores da matriz C elemento a elemento");
//--- Atravessar as linhas da matriz - objetos CRow - num loop
   for(int i=0;i<3;i++)
     {
```

```
      string com="| ";
      //--- Formar linhas a partir da matriz para o valor
      for(int j=0;j<3;j++)
        {
         //--- Obter o elemento da matriz pelo número de linha e coluna
         double element=C[i][j];// [i] - Acesso para CRow no array m_rows[] ,
                                // [j] - Operador sobrecarregado da indexação em CRow
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- Saída dos valores da linha
      Print(com);
     }
  }
//+------------------------------------------------------------------+
//| Class "Row"                                                      |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- Construtores e um destrutor
                     CRow(void)          { ArrayResize(m_array,0);    }
                     CRow(const CRow &r) { this=r;                    }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- Número de elementos na linha
   int               Size(void) const    { return(ArraySize(m_array));}
   //--- Retorna uma string com valores
   string            String(void) const;
   //--- Operador de indexação
   double            operator[](int i) const  { return(m_array[i]);   }
   //--- Operadores de atribuição
   void              operator=(const double  &array[]); // Uma array
   void              operator=(const CRow & r);         // Outro objeto CRow
   double            operator*(const CRow &o);          // Objeto CRow para multiplica
  };
//+------------------------------------------------------------------+
//| Construtor para inicializar uma linha com um array               |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- Se o array não está vazio
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- Preencher com valores
```

```
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
     }
//---
  }
//+------------------------------------------------------------------+
//| Atribuir operação para o array                                   |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- Preencher array com valores
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Operação de atribuição para CRow                                 |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- Preencher array com valores
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| Operador de multiplicação por outra linha                        |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- Verificações
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": Falha ao multiplicar duas matrizes, elas são de tamanhos di
      return(res);
     }
//--- Multiplicar arrays elemento a elemento e adicionar os produtos
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- Resultado
   return(res);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Retorno da representação da string formatada                    |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- Se o tamanho do array é maior do que zero
   int size=ArraySize(m_array);
//--- Trabalhamos apenas com números diferentes de zero dos elementos array
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- Recolher os valores para a string
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- Resultado
   return(out);
  }
//+------------------------------------------------------------------+
//| Class "Matrix"                                                   |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- Construtores e um destrutor
                     CMatrix(void);
                     CMatrix(int rows)   { ArrayResize(m_rows,rows);              }
                    ~CMatrix(void){};
   //--- Obter os tamanhos de matriz
   int               Rows()       const { return(ArraySize(m_rows));             }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- Retorna o valor da coluna na forma da linha Crow
   CRow              GetColumnAsRow(const int col_index) const;
   //--- Retorna uma string com valores de matriz
   string            String(void) const;
   //--- O operador de indexação retorna uma string pelo seu número
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));        }
   //--- Operador de adição
   CMatrix           operator+(const CMatrix &m);
   //--- Operador de multiplicação
   CMatrix           operator*(const CMatrix &m);
   //--- Operador de atribuição
   CMatrix          *operator=(const CMatrix &m);
```

© 2000-2024, MetaQuotes Ltd.

```
  };
//+------------------------------------------------------------------+
//| Um construtor padrão, cria um array de linhas de tamanho zero    |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- O número zero das linhas na matriz
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| Retorna o valor da coluna na forma de CRow                       |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- A variável para obter os valores a partir da coluna
   CRow row();
//--- O número de linhas na matriz
   int rows=Rows();
//--- Se o número de linhas maior do que zero, executar a operação
   if(rows>0)
     {
      //--- Um array para receber os valores da coluna com índice col_indez
      double array[];
      ArrayResize(array,rows);
      //--- Preenchendo o array
      for(int i=0;i<rows;i++)
        {
         //--- Verificar o número da coluna para a linha i - que podem ultrapassar os
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Erro! Número da coluna ",col_index,"> tamanho da linh
            break; // linha não inicializará o objeto
           }
         array[i]=this[i][col_index];
        }
      //--- Criar uma linha CRow baseada nos valores do array
      row=array;
     }
//--- Resultado
   return(row);
  }
//+------------------------------------------------------------------+
//| Adicionar duas matrizes                                          |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- O número de linha e colunas na matriz passada
   int cols=m.Cols();
```

```
   int rows=m.Rows();
//--- A matriz recebe os resultados da adição
   CMatrix res(rows);
//--- Os tamanhos de matriz devem coincidir
   if(cols!=Cols() || rows!=Rows())
     {
      //--- Adição impossível
      Print(__FUNCSIG__,": Falha para adicionar duas matrizes, seus tamanhos são difer
      return(res);
     }
//--- Array auxiliar
   double arr[];
   ArrayResize(arr,cols);
//--- Atravessar as linhas para adicionar
   for(int i=0;i<rows;i++)
     {
      //--- Escrever os resultados da adição das strings matriz no array
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- Colocar o array para a linha matriz
      res[i]=arr;
     }
//--- retorna o resultado da adição de matrizes
   return(res);
  }
//+------------------------------------------------------------------+
//| Multiplicação de duas matrizes                                   |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- Número de colunas da primeira matriz, número de linhas transmitidas na matriz
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- Matriz para receber o resultado da adição
   CMatrix res(rows1);
//--- Matrizes devem ser coordenadas
   if(cols1!=rows2)
     {
      //--- Multiplicação impossível
      Print(__FUNCSIG__,": Falha para multiplicar duas matrizes, formato não é compati
            "- o número de colunas no primeiro fator deveria ser igual ao número de li
      return(res);
     }
//--- Array auxiliar
   double arr[];
```

```
   ArrayResize(arr,cols1);
//--- Preencher as linhas na multiplicação da matriz
   for(int i=0;i<rows1;i++)// Atravessar as linhas
     {
      //--- Restabelecer o array recebido
      ArrayInitialize(arr,0);
      //--- Atravessar elementos na linha
      for(int k=0;k<cols1;k++)
        {
         //--- Levar valores da coluna k da matriz m para CRow
         CRow column=m.GetColumnAsRow(k);
         //--- Multiplicar duas linhas e escrever o resultado da multiplicação escala
         arr[k]=this[i]*column;
        }
      //--- colocar array arr[] na linha i-th da matriz
      res[i]=arr;
     }
//--- Retornar o produto das duas matrizes
   return(res);
  }
//+------------------------------------------------------------------+
//| Operação de atribuição                                           |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- Preencher e defineir o número de linhas
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- Preencher nossas linhas com valores das linhas da matriz anterior
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| Representação string da matriz                                   |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- Formar string por string
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- Resultado
   return(out);
  }
```

Também Veja

Sobrecarga, Operações Aritméticas, Sobrecarga de Função, Regras de Precedência

© 2000-2024, MetaQuotes Ltd.

Descrição de Funções Externas

As funções externas definidas em outro módulo devem ser explicitamente descritas. A descrição inclui o tipo retornado, o nome da função e uma série dos parâmetros de entrada com os seus tipos. A ausência de tal descrição pode levar a erros ao compilar, construir e executar um programa. Ao descrever um objeto externo, use a palavra-chave *#import* indicando o módulo.

Exemplos:

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

Com a ajuda do import, é fácil descrever funções que são chamadas de DLL externas ou de bibliotecas (libraries) EX5 compiladas. Bibliotecas (libraries) EX5 são arquivos ex5 compilados, que têm a propriedade library. Somente funções descritas com o modificador export pode ser importadas de bibliotecas (libraries) EX5.

Lembre-se de que as bibliotecas DLL e EX5 devem ter nomes diferentes (independentemente dos diretórios em que estão localizados) se forem importados juntos. Todas as funções importadas têm a resolução de escopo correspondente à biblioteca "file name".

Exemplo:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // Chamada do método de classe
      Print(::GetLastError());         // chamar a função MQL5
      Print(kernel32::GetLastError()); // Chamada da função de biblioteca DLL a parti
      Print(lib::GetLastError());      // Chamada da função da biblioteca EX5 lib.ex5
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
  }
```

© 2000-2024, MetaQuotes Ltd.

Também Veja

Sobrecarga (Overload), Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Exportação de Funções

Uma função declarada em um programa MQL5 com o pós-modificador *export* pode ser usada em um outro programa MQL5. Tal função é chamada exportável, e ela pode ser chamada de outros programas após compilação.

```
int Function() export
  {
  }
```

Este modificador ordena o compilador a adicionar a função na tabela de funções EX5 exportada por este arquivo ex5. Somente funções com tal modificador podem ser acessíveis ("visíveis") a partir de outros programas MQL5.

A propriedade library diz ao compilador que o arquivo-EX5 será uma biblioteca (library), e o compilador o exibirá no cabeçalho da EX5.

Todas as função que são planejadas a serem exportáveis devem ser marcadas com o modificador *export*.

Também Veja

Sobrecarga (Overload), Funções Virtuais, Polimorfismo

© 2000-2024, MetaQuotes Ltd.

Funções de Manipulação de Evento

A linguagem MQL5 fornece processamento de alguns eventos pré-definidos. Funções para manipulação destes eventos devem ser definidas em um programa MQL5; nome da função, tipo de retorno, composição dos parâmetros (se existir algum) e seus tipos devem estar rigorosamente em conformidade com a descrição da função de handler (manipulador) de evento.

O handler (manipulador) de evento do terminal cliente identifica funções, manipulando este ou aquela evento, pelo tipo de valor de retorno e tipo de parâmetros. Se outros parâmetros, que não correspondem às descrições abaixo, são especificadas para uma função correspondente, ou outro tipo de retorno é indicado para ela, tal função não será usada como um handler (manipulador) de evento.

OnStart

A função OnStart() é o handler (manipulador) do evento Start ,que é automaticamente gerado **somente** para execuções de **scripts**. Ela dever ser do tipo void, sem parâmetros:

```
void OnStart();
```

Para a função OnStart(), o tipo de retorno int pode ser especificado.

OnInit

A função OnInit() é o handler (manipulador) do evento Init. Ela deve ser do tipo void ou int, sem parâmetros:

```
void OnInit();
```

O evento Init é gerado imediatamente após um Expert Advisor ou um indicador ser baixado; este evento não é gerado para scripts. A função OnInit() é usada para inicialização. Se OnInit() tiver o tipo int de valor de retorno, o código de retorno não-zero significa inicialização sem sucesso e é gerado o evento Deinit com o código do motivo da desinicialização REASON_INITFAILED.

Ao retornar o valor INIT_FAILED o Expert Advisor será retirado à força do gráfico.

Ao retornar o valor INIT_FAILED o indicador não será removido do gráfico. Nesse caso, o indicador restante no gráfico estará inoperante — os manipuladores de eventos não são chamados no indicador.

Para otimizar os parâmetros de entrada de um Expert Advisor, é recomendado usar valores da enumeração ENUM_INIT_RETCODE como código de retorno. Esses valores são usados para organizar o curso da otimização, incluindo a seleção dos mais apropriados agentes de teste. Durante a inicialização de um Expert Advisor, antes do início do teste, você pode solicitar informação sobre a configuração e recursos de um agente (o número de cores, quantidade de memória livre, etc) usando a função TerminalInfoInteger(). Baseado nestas informações obtidas, pode-se tanto permitir usar este agente de teste, ou rejeitar usá-lo durante a otimização deste Expert Advisor.

ENUM_INIT_RETCODE

Identificador	Descrição
INIT_SUCCEEDED	Inicialização bem sucedida, teste do Expert Advisor pode continuar.

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	Este código significa o mesmo que um valor null = o Expert Advisor foi inicializado com sucesso no Provador de Estratégia.
INIT_FAILED	Inicialização com falha; não existe motivo para continuar testando por cause de erros fatais. Por exemplo, falha em criar um indicador que é requerido para o funcionamento do Expert Advisor. Este valor de retorno significa o mesmo que um valor diferente de zero - inicialização do Expert Advisor pelo Provador de Estratégia falhou.
INIT_PARAMETERS_INCORRECT	Este valor significa a incorreta definição da entrada de parâmetros. A seqüência de resultado contendo o código de retorno é destacado em vermelho na tabela de otimização geral. O teste para o dado conjunto de parâmetros do Expert Advisor não será executado, o agente é liberado para receber uma nova tarefa. Depois de receber este valor, o testador de estratégia com segurança não vai passar essa tarefa para que outros agentes tentem novamente.
INIT_AGENT_NOT_SUITABLE	Nenhum erro durante a inicialização, mas por alguma razão o agente não é adequado para o teste. Por exemplo, ausência de memória suficiente, sem support ao OpenCL, etc. Após este código de retorno, o agente não receberá tarefas até o fim desta otimização.

A função OnInit() do tipo void sempre indica inicialização bem sucedida.

OnDeinit

A função OnDeinit() é chamada durante a desinicialização e é o handler (manipulador) do evento Deinit. Ela deve ser declarada com o tipo void e ter um parâmetro do tipo const int, que contém o código do motivo da desinicialização. Se um tipo diferente é declarado, o compilador gerará um aviso e a função não será chamada. Para scripts o evento Deinit não é gerado e portanto a função OnDeInit() não pode ser usada em scripts.

```
void OnDeinit(const int reason);
```

O evento Deinit é gerado para Expert Advisors e indicadores nos seguintes casos:

- antes de uma reinicialização devido à mudança de ativo (symbol) ou período do gráfico no qual o programa MQL5 está anexado;
- antes de uma reinicialização devido à mudança de parâmetros de entrada;
- antes de descarregar o programa MQL5.

OnTick

© 2000-2024, MetaQuotes Ltd.

O evento NewTick é gerado para **Expert Advisors somente** quanto um novo preço (tick) para um ativo é recebido pelo gráfico no qual o Expert Advisor está anexado. É inútil definir a função OnTick() num indicador personalizado ou script, porque o evento NewTick não é gerado para eles.

O evento Tick é gerado apenas para Expert Advisors, mas isso não significa que Expert Advisors requer a função OnTick(), uma vez que não são apenas os eventos NewTick gerados para Expert Advisors, mas também são gerados os eventos de Timer, BookEvent e ChartEvent. Ela deve ser declarada com o tipo void, sem parâmetros:

```
void OnTick();
```

OnTimer

A função OnTimer() é chamada quando o evento Timer ocorre, que é gerado pelo timer do sistema somente para Expert Advisors e indicadores - ela não pode ser usada em scprits. A freqüência de ocorrência do evento é definida na subscrição de notificações deste evento através da função EventSetTimer().

Você pode desfazer a subscrição de receber eventos de timer para um Expert Advisor particular usando a função EventKillTimer(). A função de ser definida com o tipo void, sem parâmetros:

```
void OnTimer();
```

É recomendável chamar a função EventSetTimer() uma vez na função OnInit(), e a função EventKillTimer() deve ser chamada uma vez em OnDeinit().

Todo Expert Advisor, assim como todo indicador funciona com seu próprio timer e recebe eventos apenas a partir dele. Tão logo um programa MQL5 para de funcionar, o timer é destruído de forma forçada, se ele foi criado mas não desabilitado pela função EventKillTimer().

OnTrade

A função é chamada quando o evento Trade ocorre, que surge quando você muda a lista de ordens postadas e posições abertas, o histórico de ordens e histórico de operações (deals). Quando uma atividade de negociação (trade) é realizada (abertura de ordem pendente, abertura/fechamento de posição, definição de stop, disparo de ordem pendente, etc.) o histórico de ordens e operações (deals) e/ou a lista de posições e ordens correntes são por conseqüência alterados.

```
void OnTrade();
```

Os usuários devem implementar de forma independente no código a verificação do estado de uma conta de negociação quanto tal evento é recebido (se isto é requerido pelas condições da estratégia de negócio). Se a chamada da função OrderSend() foi concluída com sucesso e retornou um valor de true, isso significa que o servidor de negociação postou a ordem na fila de execução e atribuiu um número de bilhetagem (ticket number) nele. Tão logo o servidor processe esta ordem, o evento Trade será gerado. E se um usuário lembrar valor da bilhetagem (ticket), ele/ela será capaz de descobrir o que aconteceu com a ordem usando este valor na função OnTrade().

OnTradeTransaction

Ao realizar algumas ações específicas em uma conta de negociação, seu estado muda. Tais ações includem:

© 2000-2024, MetaQuotes Ltd.

- Envio de uma solicitação de negociação a partir de algum aplicativo MQL5 no terminal usando as funções OrderSend e OrderSendAsync e sua posterior execução.
- Envio de uma solicitação de negociação por meio da interface gráfica do terminal e sua posterior execução.
- Ativação de ordens pendentes e ordens de stop no servidor.
- Realização de operações no lado de um servidor de negociação.

As seguintes transações de negociação são realizadas como resultado destas ações:

- tratamento de uma solicitação de negociação
- mudança de ordens de abertura
- mudança de histórico de ordens
- mudança de histórico de operações (deals)
- mudança de posições

Por exemplo, ao enviar uma ordem de compra de mercado, ela é tratada, uma ordem de compra apropriada é criada para a conta, a ordem é então executada e removida da lista de ordens em aberto, e então ela é adicionada ao histórico de ordens, uma apropriada operação (deal) é adicionada ao histórico e uma nova posição é criada. Todas estas ações são transações de negociação. A chegada de tal transação no terminal é um evento TradeTransaction. Ele chama o handler (manipulador) de evento OnTradeTransaction.

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // estrutura das transações de negócios
   const MqlTradeRequest&        request,      // estrutura solicitada
   const MqlTradeResult&         result        // resultado da estrutura
   );
```

O handler (manipulador) contém três parâmetros:

- **trans** - este parâmetro obtém a estrutura MqlTradeTransaction descrevendo uma transação de negociação aplicada a uma conta de negócio.
- **request** - este parâmetro obtém a estrutura MqlTradeRequest descrevendo uma solicitação de negócio;
- **result** - este parâmetro obtém a estrutura MqlTradeResult descrevendo o resultado da execução de uma solicitação de negociação.

Os últimos dois parâmetros, **request** e **result**, são preenchidos por valores somente para uma transação de tipo TRADE_TRANSACTION_REQUEST, dados sobre uma transação podem ser recebidos a partir do parâmetro do *tipo* da variável **trans**. Note que neste caso, o campo *request_id* na variável **result** contém o ID da **solicitação** de negócio, após a execução da transação de negociação, descrita na variável **trans**, ter sido realizada. O identificador da solicitação (Request ID) permite associar a ação realizada (chamada de funções OrderSend ou OrderSendAsync) com o resultado da ação enviado para OnTradeTransaction().

Uma solicitação de negociação manualmente enviada a partir do terminal ou via funções OrderSend()/OrderSendAsync() podem gerar várias transações consecutivas no servidor de negócios. A prioridade de chegada dessas transações no terminal não é garantida. Assim, você não deve esperar que um grupo de transações chegará após um outro grupo ao desenvolver seu algoritmo de negociação.

© 2000-2024, MetaQuotes Ltd.

- Todos os tipo de transações de negociação são descritas na enumeração ENUM_TRADE_TRANSACTION_TYPE.
- A estrutura MqlTradeTransaction descrevendo um transação de negociação é preenchida de diferentes formas dependendo do tipo de transação. Por exemplo, somente o campo de tipo (tipo de transação de negociação) deve ser analisado para transações do tipo TRADE_TRANSACTION_REQUEST. O segundo e terceiro parâmetros (request e result) da função OnTradeTransaction deve ser analisado para dados adicionais. Para informações adicionais, veja Estrutura de uma Transação de Negociação.
- Uma descrição de transação de negociação não entrega todas as informações disponíveis relativas a ordens, operações (deals) e posições (por exemplo, comentários). As funções OrderGet*, HistoryOrderGet*, HistoryDealGet* e PositionGet* devem ser usadas para obter informações adicionais.

Após aplicar transações de negociação em uma conta de cliente, elas são consistentemente postadas na fila de transações de negócio do terminal, a partir da qual são consistentemente enviados para o ponto de entrada OnTradeTransaction na ordem de chegada no terminal.

Ao tratar transações de negociação por um Expert Advisor usando o handler OnTradeTransaction (Manipulador sobre Transação de Comércio), o terminal continua manipulando as transações de negociação recém chegadas. Portanto, o estado de uma conta de negociação pode mudar durante uma operação OnTradeTransaction. Por exemplo, enquanto um programa MQL5 manipula um evento para adicionar uma nova ordem, ela pode ser executada, deletada da lista das abertas e movida para o histórico. Mais adiante, o aplicativo será notificado destes eventos.

O comprimento da fila de transações compreende 1024 elementos. Se OnTradeTransaction tratar uma nova transação por muito tempo, as transações mais antigas na fila podem ser substituídas pelas novas.

- De forma geral, não existe um proporção precisa entre o número de chamadas de OnTrade e OnTradeTransactions. Uma chamada OnTrade corresponde a uma ou várias chamadas OnTradeTransactions.
- OnTrade é chamada após apropriadas chamadas OnTradeTransaction.

OnTester

A função OnTester() é o handler (manipulador) do evento Tester que é automaticamente gerado após um teste de histórico de um Expert Advisor no intervalo escolhido ter terminado. A função deve estar definida com o tipo double, sem parâmetros:

```
double OnTester();
```

A função é chamada logo antes da chamada de OnDeinit() e tem o mesmo tipo do valor de retorno - double. OnTester() pode ser usado apenas no teste de Expert Advisors. Seu principal propósito é calcular um certo valor que é usado como o critério max customizado na otimização genética de parâmetros de entrada.

Na otimização genética, a ordenação descendente é aplica aos resultados de uma geração. Isto é, do ponto de vista do critério de otimização, os melhores resultados são aqueles com o maiores valores (os valores do critério de otimização max customizado retornados pela função OnTester são levados

© 2000-2024, MetaQuotes Ltd.

em consideração). Em tal ordenação, os piores valores são posicionados no final e posteriormente jogados fora e não participam na formação da nova geração.

OnTesterInit

A função OnTesterInit() é o handler(manipulador) do evento TesterInit, que é automaticamente gerado antes de iniciar a otimização do Expert Advisor no Provador de Estratégia. A função deve ser definida com o tipo void. Ele não tem parâmetros:

```
void OnTesterInit();
```

Com o início da otimização, um Expert Advisor com o handler (manipulador) OnTesterInit() ou OnTesterPass() é automaticamente carregado em um gráfico separado do terminal com o ativo e período especificados no Provador de Estratégia, e recebe o evento TesterInit. A função é usada para inicializar o Expert Advisor antes de iniciar a otimização para posterior processamento dos resultados da otimização.

OnTesterPass

A função OnTesterPass() é um handler (manipulador) do evento TesterPass, que é automaticamente gerado quanto um plano é recebido durando a otimização de um Expert Advisor no Provador de Estratégia. A função deve ser definida com o tipo void. Ele não tem parâmetros:

```
void OnTesterPass();
```

Um Expert Advisor com o handler (manipulador) OnTesterPass() é automaticamente carregado em um gráfico separado do terminal com o ativo/período especificados para teste, e obtém eventos TesterPass quando um plano é recebido durante uma otimização. A função é usada para tratamento dinâmico dos resultados de otimização "no local" sem precisar esperar pela sua conclusão. Planos são adicionados usando a função FrameAdd(), que pode ser chamada após o fim de um passo único no handler (manipulador) OnTester() .

OnTesterDeinit

OnTesterDeinit() é um handler (manipulador) do TesterDeinit, que é automaticamente gerada após o fim da optimização de um Expert Advisor no Provador de Estratégia. A função deve ser definida com o tipo void. Ele não tem parâmetros:

```
void OnTesterDeinit();
```

Um Expert Advisor com o handler (manipulador) TesterDeinit() é automaticamente carregada em um gráfico no início da otimização, e recebe TesterDeinit após sua conclusão. A função é usada para um processamento final de todos os resultados da otimização.

OnBookEvent

A função OnBookEvent() é o handler (manipulador) do BookEvent. BookEvent é gerado para Expert Advisors e indicadores somente quando a Profundidade do Mercado muda. Ela deve do tipo void e ter um parâmetro do tipo string:

```
void OnBookEvent (const string& symbol);
```

© 2000-2024, MetaQuotes Ltd.

Para receber eventos BookEvent para qualquer ativo (symbol), você apenas precisa fazer uma pré-subscrição pra receber eventos para este ativo usando a função MarketBookAdd(). A fim de desfazer a subscrição de recebimento de eventos BookEvent para um particular ativo, chame MarketBookRelease().

Diferente de outros eventos, o evento BookEvent é por difusão (broadcast). Isso significa que se um Expert Advisor subscreve para receber eventos BookEvent usando MarketBookAdd, todos os outros Experts Advisors que tem o handler (manipulador) OnBookEvent() receberão este evento. É portanto necessário analisar o nome do ativo, que é passado para o handler (manipulador) através dos parâmetros *const string& symbol*.

OnChartEvent

OnChartEvent() é o handler (manipulador) de um grupo de eventos ChartEvent:

- CHARTEVENT_KEYDOWN — evento de uma teclada, quando a janela do gráfico está com foco;
- CHARTEVENT_MOUSE_MOVE — eventos de movimento de mouse e eventos de click de mouse (se CHART_EVENT_MOUSE_MOVE=true é definido para o gráfico);
- CHARTEVENT_OBJECT_CREATE — evento de criação de objeto gráfico (se CHART_EVENT_OBJECT_CREATE=true é definido para o gráfico);
- CHARTEVENT_OBJECT_CHANGE — evento de mudança de um propriedade de objeto via janela diálogo de propriedades;
- CHARTEVENT_OBJECT_DELETE — evento de exclusão de objeto gráfico (se CHART_EVENT_OBJECT_DELETE=true é definido para o gráfico);
- CHARTEVENT_CLICK — evento de um click de mouse no gráfico;
- CHARTEVENT_OBJECT_CLICK — evento de um click de mouse em um objeto gráfico pertencente ao gráfico;
- CHARTEVENT_OBJECT_DRAG — evento de um movimento de objeto gráfico usando o mouse;
- CHARTEVENT_OBJECT_ENDEDIT — evento da edição de texto finalizada na caixa de entrada do objeto gráfico LabelEdit;
- CHARTEVENT_CHART_CHANGE — evento de mudanças de gráfico;
- CHARTEVENT_CUSTOM+n — ID do evento do usuário, onde n está na faixa de 0 a 65535.
- CHARTEVENT_CUSTOM_LAST — o último ID aceitável de um evento customizado (CHARTEVENT_CUSTOM+65535).

A função pode ser chamada somente em Expert Advisors e indicadores. A função deve ser de tipo void com 4 parâmetros:

```
void OnChartEvent(const int id,          // Evento ID
                  const long& lparam,    // Parâmetro de evento de tipo long
                  const double& dparam,  // Parâmetro de evento de tipo double
                  const string& sparam   // Parâmetro de evento de tipo string
  );
```

Para cada tipo de evento, os parâmetros de entrada da função OnChartEvent() têm valores definidos que são requeridos para o processamento deste evento. Os eventos e valores passados através destes parâmetros são listados na tabela abaixo.

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de uma teclada	CHARTEVENT_KEYDOWN	código de uma tecla pressionada	Repita a contagem (o número de vezes que a tecla é repetida como um resultado de que o usuário pressiona a tecla)	O valor da string de uma pequena máscara de descrever o estado de botões do teclado
Eventos de mouse (se a propriedade CHART_EVENT_MOUSE_MOVE=true está definida para o gráfico)	CHARTEVENT_MOUSE_MOVE	a coordenada X	a coordenada Y	O valor de string de uma máscara de bites descrevendo o estado de botões de mouse
Evento de criação de objeto gráfico (se CHART_EVENT_OBJECT_CREATE=true, então é definido para o gráfico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome do objeto gráfico criado
Evento de mudança de uma propriedade de objeto via janela de diálogo de propriedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nome do objeto gráfico modificado
Evento de exclusão de objeto gráfico (se CHART_EVENT_OBJECT_DELETE=true está definido para o gráfico)	CHARTEVENT_OBJECT_DELETE	—	—	Nome do objeto gráfico excluído
Evento de um click de mouse no gráfico	CHARTEVENT_CLICK	a coordenada X	a coordenada Y	—

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de um click de mouse num objeto gráfico pertencente ao gráfico	CHARTEVENT_OBJECT_CLICK	a coordenada X	a coordenada Y	Nome do objeto gráfico, na qual o evento ocorreu
Evento de um objeto gráfico arrastado usando o mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome do objeto gráfico movido
Evento da edição de texto finalizada na caixa de entrada do objeto gráfico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome do objeto gráfico LabelEdit, cuja edição de texto foi concluída
Evento de mudanças de gráfico	CHARTEVENT_CHART_CHANGE	—	—	—
ID do evento de usuário sob N número	CHARTEVENT_CUSTOM+N	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()

OnCalculate

A função OnCalculate() é chamada somente em indicadores customizados quando é necessário calcular os valores do indicador pelo evento Calculate. Isso geralmente acontece quando um novo preço (tick) é recebido para o ativo, de cujo indicador é calculado. Não é necessário que este indicador esteja anexado a qualquer gráfico de preço deste ativo.

A função OnCalculate() deve retornar um tipo int. Existem duas possíveis definições. Dentro de um indicador você não pode usar ambas as versões da função.

A primeira forma é destinado para aqueles indicadores que podem ser calculados com um único buffer de dados. Um exemplo de tal indicador é a Média Móvel Customizada (Custom Moving Average).

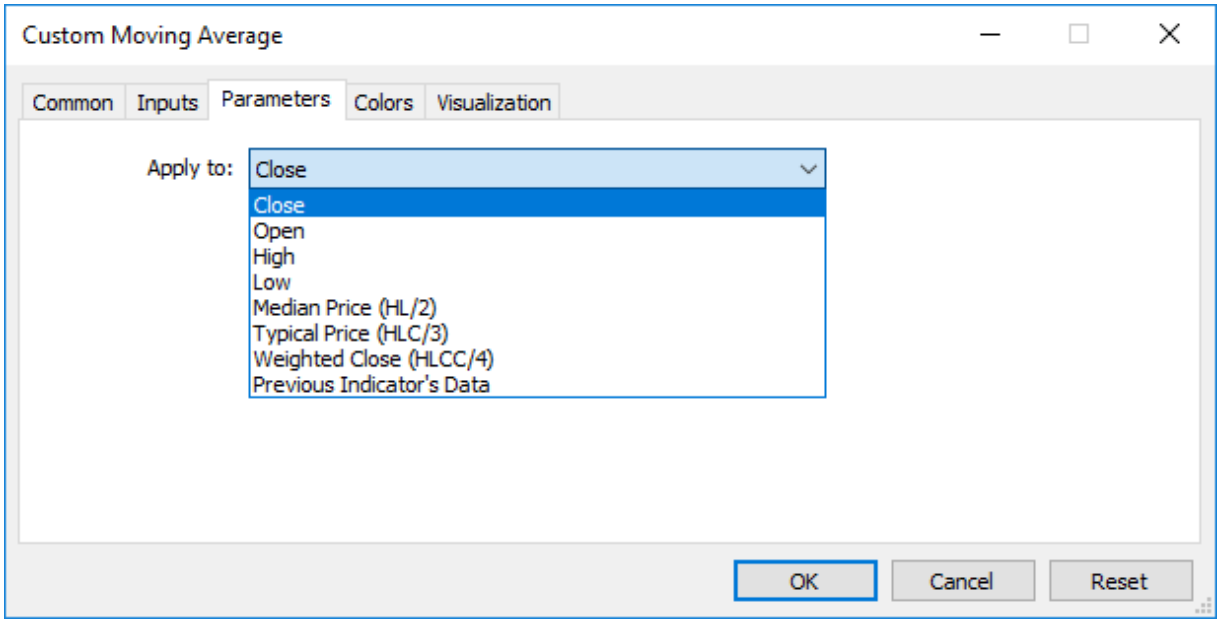
```
int OnCalculate (const int rates_total,      // tamanho do array price[]
                 const int prev_calculated,  // barras tratadas na chamada anterior
                 const int begin,            // a partir de onde começam os dados sig
                 const double& price[]       // array a ser calculado
   );
```

Assim como o array price[], uma das série de preço ou um buffer calculado de algum indicador pode ser passado. Para determinar a direção da indexação no array price[], chame ArrayGetAsSeries(). A

© 2000-2024, MetaQuotes Ltd.

fim de não depender de valores default, você deve incondicionalmente chamar a função ArraySetAsSeries() para aqueles arrays que você espera utilizar.

Uma série de tempo necessária ou um indicador, para ser usado como o array price[], deve ser selecionado pelo usuário na guia "Parâmetros" ao iniciar o indicador. Para fazer isso, você deve especificar o necessário item no lista drop-down do campo "Aplicar a".



Para receber valores de um indicador customizado a partir outros programas mql5, a função iCustom() é usada, que retorna o manuseio do indicador para operações subseqüentes. Você pode também especificar o price[] array apropriado ou o manuseio de outro indicador. Este parâmetro deve ser transmitido por último na lista de variáveis de entrada do indicador customizado.

Exemplo:

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("Pode abrir ou não o arquivo EX5 '"+terminal_path+"\\MQL5\\Indicators\\"+'
  }
```

Neste exemplo, o último parâmetro passado é o valor PRICE_TYPICAL (da enumeração ENUM_APPLIED_PRICE), que indica que o indicador customizado será construído baseado em preços típicos obtidos como (High+Low+Close)/3. Se este parâmetro não for especificado, o indicador é construído baseado em valores de PRICE_CLOSE, isto é, preços de fechamento de cada barra.

Outro exemplo que mostra a passagem de um handler (manipulador) de indicador como o último parâmetro para especificar o array price[], é dado na descrição da função iCustom().

© 2000-2024, MetaQuotes Ltd.

A segunda forma é destinada para todos os outros indicadores, na qual mais de uma série de tempo é usada nos cálculos.

```
int OnCalculate (const int rates_total,      // tamanho da série de preços de entrada
                 const int prev_calculated,  // barras tratadas na chamada anterior
                 const datetime& time[],     // Hora
                 const double& open[],       // Open (abertura)
                 const double& high[],       // High (máximo)
                 const double& low[],        // Low (mínimo)
                 const double& close[],      // Close (fechamento)
                 const long& tick_volume[],  // Volume de Tick
                 const long& volume[],       // Volume Real
                 const int& spread[]         // Spread
   );
```

Os parâmetros open[], high[], low[] and close[] contém os arrays com preços de abertura, preços de máximo e mínimo e preços de fechamento da janela de tempo corrente. O parâmetro time[] contém um array com valores de hora de abertura, o parâmetro spread[] tem um array contendo o histórico de spreads (se algum spread é fornecido para o ativo negociado). Os parâmetros volume[] e tick_volume[] contêm o histórico de volume de negociação e tick, respectivamente.

Par determinar a direção de indexação de time[], open[], high[], low[], close[], tick_volume[], volume[] e spread[], chame ArrayGetAsSeries(). A fim de não depender de valores default, você deve incondicionalmente chamar a função ArraySetAsSeries() para aqueles arrays que você esperar utilizar.

O primeiro parâmetro rates_total contém o número de barras disponíveis no indicador para cálculo, e corresponde ao número de barras disponíveis no gráfico.

Devemos notat a conexão entre o valor de retorno de OnCalculate() e o segundo parâmetro de entrada prev_calculated. Durante a chamada da função, o parâmetro prev_calculated contém um valor **retornado** pelo OnCalculate() durante a chamada **anterior**. Isso permite que algoritmos eficientes calculem o indicador customizado de forma a evitar cálculos repetidos naquelas barras que não tiveram mudança deste a execução anterior desta função.

Para isso, é geralmente suficiente retornar o valor do parâmetro rates_total, que contém o número de barras da chamada corrente da função. Se desde a última chamada da função OnCalculate() os dados de preço mudarem (um histórico mais antigo baixado ou brancos no histórico preenchidos), o valor do parâmetro de entrada prev_calculated será definido para zero pelo terminal.

Observação: se OnCalculate retornar zero, então os valores do indicador não são mostrados na Janela de Dados do terminal cliente.

Para entender isso melhor, seria útil iniciar o indicador, cujo código está anexado abaixo.

Exemplo indicador:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar Linha
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers do indicador
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- Obtenção do número de barras disponíveis para o ativo corrente e período do grá
   int bars=Bars(Symbol(),0);
   Print("Bars = ",bars,", rates_total = ",rates_total,",  prev_calculated = ",prev_ca
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Também Veja

Programas em Execução, Eventos do Terminal Cliente, Trabalhando com Eventos

© 2000-2024, MetaQuotes Ltd.

Variáveis

Declarando Variáveis

Variáveis devem ser declaradas antes de serem usadas. Nomes únicos são usados para identificar variáveis. Para declarar uma variável, você deve especificar o seu tipo e nome único. Declaração de variável não é um operador.

Os tipos simples são:

- char, short, int, long, uchar, ushort, uint, ulong - inteiros;
- color - inteiro representando a cor-RGB (red, green, blue);
- datetime - a data e hora, um inteiro sem sinal (unsigned) contendo o número de segundos desde a 0 hora de 1 de Janeiro de 1970;
- bool - valores booleanos *true* e *false*;
- double - número de ponto flutuante de dupla precisão;
- float - número de ponto flutuante de precisão simples;
- string - cadeias de caracteres.

Exemplos:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Tipos compostos ou complexos:

Estruturas são tipos de dados compostos construídas usando outros tipos.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // Variável previamente declarada da estrutura MyTime
```

Não se pode declarar variáveis do tipo estrutura até que se declare a estrutura.

Arrays

Array (ou matriz) é uma seqüência indexada de dados de tipos idênticos:

```
int    a[50];       // Array de uma dimensão de 50 inteiros.
double m[7][50];    // Array de duas dimensões de sete arrays,
                    // cada um deles constituído de 50 números.
MyTime t[100];      // Array contendo elementos como MyTime
```

© 2000-2024, MetaQuotes Ltd.

Somente um inteiro pode ser um índice de array. Não são permitidos arrays com mais que quatro dimensões. A numeração de elementos de um array começa com 0. O último elemento de um array de uma dimensão tem o número do índice que é 1 a menos que o tamanho do array. Isso significa que chamar pelo último elemento de um array constituído de 50 aparecerá como a[49]. O mesmo acontece a arrays multidimensionais: uma dimensão é indexada de 0 ao tamanho da dimensão menos 1. O último elemento de um array de duas dimensões aparecerá como m[6][49].

Arrays estáticos não podem ser representados como séries de tempo, isto é, a funçãoArraySetAsSeries() não pode ser aplicada a eles, esta função define o acesso aos elementos de um array do fim para o começo. Se você desejar ter acesso a um array da mesma forma que uma série de tempo, use o objeto de array dinâmico.

Se houver uma tentativa de acesso fora da faixa do array, a execução do subsistema gerará um erro crítico e o programa será interrompido.

Métodos embutidos para trabalhar usando arrays

Para trabalhar com arrays, você pode usar as funções da seção Operações de arrays, bem como métodos embutidos:

Método	Análogo	Descrição
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Preenche o array com o valor especificado
void array.Free();	ArrayFree	Libera o buffer dinâmico do array e define o tamanho da dimensão zero como 0 (zero)
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Estabelece um novo tamanho na primeira dimensão do array
int array.Print();	ArrayPrint	Imprime os valores de um array de tipo simples no log
int array.Size(const int range=-1);	ArraySize, ArrayRange	Retorna o número de elementos em todo o array (range=-1) ou na dimensão especificada do array
bool array.IsDynamic();	ArrayIsDynamic	Verifica se o array é dinâmico
bool array.IsIndicatorBuffer();		Verifica se o array é um buffer indicador
bool array.IsSeries();	ArrayIsSeries	Verifica se o array é uma série temporal
bool array.AsSeries();	ArrayGetAsSeries	Verifica a direção de indexação do array
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Define a direção da indexação no array

© 2000-2024, MetaQuotes Ltd.

Método	Análogo	Descrição
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Copia os valores de um array em outro array
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	Retorna o resultado de uma comparação de dois arrays de tipos simples ou de estruturas personalizadas
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Insere na matriz de destino o número especificado de elementos da matriz de origem, iniciando no índice especificado
int array.Remove(const int start_pos, const int count);	ArrayRemove	Elimina o número especificado de elementos do array a partir do índice especificado
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Desdobra o número especificado de elementos do array, começando pelo índice especificado
bool array.Swap(array& arr[]);	ArraySwap	Troca de conteúdo com outro array dinâmico do mesmo tipo
void array.Sort(sort_function);	ArraySort	Classificação de arrays numéricos por primeira dimensão
int array.Search(scalar value, search_function);	ArrayBsearch	Retorna o índice do primeiro elemento encontrado na primeira dimensão do array
int array.Find((scalar value, search_function);		Procura o array usando a função passada e retorna o índice do primeiro elemento encontrado
array array.Select(scalar value, search_function);		Procura o array usando a função passada e retorna o array com todos os elementos encontrados

Especificadores de Acesso

Especificadores de acesso definem como o compilador pode acessar variáveis, membros de estruturas ou classes.

O especificador const declara uma variável como uma constante e não permite modificar esta variável durante o tempo de execução. Uma inicialização simples de uma variável é permitida durante sua declaração.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
int OnCalculate (const int rates_total,      // tamanho do array price[]
                 const int prev_calculated,  // barras tratadas na chamada anterior
                 const int begin,            // a partir de onde começam os dados sig
                 const double& price[]       // array a ser calculado
   );
```

Para acessar membros de estruturas e classes use os seguintes qualificadores:

- public - permite acesso irrestrito às variáveis ou métodos da classe
- protected - permite acesso a partir dos métodos desta classe, bem como a partir dos métodos de classes publicamente herdadas. Outro acesso é impossível;
- private - permite acesso às variáveis e métodos da classe somente a partir dos métodos da mesma classe.
- virtual - aplica-se apenas a métodos de classe (não a métodos de estrutura) e diz ao compilador que este método deve se colocado na tabela de funções virtuais da classe.

Classes de Armazenamento

Existem três classes de armazenamento: static, input e extern. Estes modificadores de classe de armazenamento explicitamente indicam ao compilador que as variáveis correspondentes estão distribuídas em uma área pré-alocada de memória, que é chamada de global pool. Além disso, estes modificadores indicam um processamento especial dos dados da variável. Se uma variável declarada em nível local não for uma variável estática, a memória para tal variável é alocada automaticamente em uma pilha de programa. A liberação de memória alocada para um array não estático (non-static) é também realizada automaticamente quando se passa além da área de visibilidade do bloco em que o array está declarado.

Veja também

Tipos de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criação e Exclusão de Objetos, Membros Estáticos de uma Classe

© 2000-2024, MetaQuotes Ltd.

Variáveis Locais

Uma variável declarada dentro de uma função é local. O escopo de uma variável local é limitado a região dentro da função na qual a variável está declarada. Uma variável local pode ser inicializada pelo resultado de qualquer expressão. Toda chamada da função inicializa uma variável local. Variáveis locais são armazenadas na área de memória da correspondente função.

Exemplo:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

Escopo de uma variável é uma parte de um programa, na qual uma variável pode ser referenciada. Variáveis declaradas dentro de um bloco (a nível interno), possuem o bloco como seu escopo. O escopo de bloco começa com a declaração da variável e termina com a chave direita final.

Variáveis locais declaradas no começo de uma função também têm escopo de bloco, bem como parâmetros de função, que são variáveis locais. Qualquer bloco pode conter declarações de variáveis. Se blocos estiverem aninhados e um identificador no bloco externo tiver o mesmo nome que um identificador no bloco interno, o identificador no bloco externo é oculto, até que a operação do bloco interno seja finalizado.

Exemplo:

```
void OnStart()
  {
//---
   int i=5;       // variável local da função
     {
      int i=10;  // variável de função
      Print("Dentro do bloco i = ",i); // resultado é i=10;
     }
   Print("Fora do bloco i = ",i);  // resultado é i=5;
  }
```

Isso significa que enquanto o bloco interno está rodando, ele vê os valores de seus próprios identificadores locais, e não os valores dos identificadores com nomes idênticos no bloco externo.

Exemplo:

```
void OnStart()
  {
//---
   int i=5;       // variável local da função
   for(int i=0;i<3;i++)
      Print("Dentro for i = ",i);
   Print("Fora do bloco i = ",i);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
/* Resultado da execução
Dentro for i = 0
Dentro for i = 1
Dentro for i = 2
Fora do bloco i = 5
*/
```

Variáveis locais declaradas como estáticas (static) têm escopo de bloco, apesar do fato de elas existirem desde o início do programa.

Pilha (Stack)

Em todo programa MQL5, uma área especial de memória chamada pilha é alocada para armazenar variáveis locais de funções que são criadas automaticamente. Só uma pilha é alocada para todas as funções e, por padrão, seu tamanho é de 1 MB. Em Advisors e scripts, o tamanho da pilha pode ser gerenciado usando a diretiva de compilação #property stacksize (define o tamanho da pilha em bytes), por padrão, para eles são alocados 8MB.

Variáveis locais estáticas (static) são armazenadas no mesmo lugar onde outras variáveis estáticas e globais são armazenadas - em uma área especial de memória, que existe separadamente da pilha. Variáveis criadas dinamicamente também usam uma área de memória separada da pilha.

A cada chamada de função, um lugar na pilha é alocado para variáveis internas não estáticas. Após sair da função, a memória fica disponível para uso novamente.

Se da primeira função, uma segunda função for chamada, então a segunda função ocupa o tamanho requerido da memória restante da pilha para suas variáveis. Portanto, ao usar funções incluídas, a memória da pilha será seqüencialmente ocupada por cada função. Isso pode levar a uma falta de memória durante uma das chamadas de função, tal situação é chamada de estouro de pilha (stack overflow).

Portanto, para grandes dados é melhor usar memória dinâmica - ao entrar em uma função, aloque a memória que é requerida para as necessidades locais (new, ArrayResize()), e ao sair da função, libere a memória (delete, ArrayFree()).

Também Veja

Tipos de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Parâmetros Formais

Parâmetros passados para a função são locais. O escopo é de bloco de função. Parâmetros formais devem ter nomes diferentes de variáveis externas e variáveis locais definidas dentro de uma função. Alguns valores podem ser atribuídos a parâmetros formais no bloco de função. Se um parâmetro formal é declarado com o modificador const , seu valor não pode ser modificado dentro da função.

Exemplo:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

Parâmetros formais podem ser inicializados por constantes. Neste caso, o valor de inicialização é considerado como o valor default (padrão). Parâmetros, próximos ao parâmetro inicializado, também devem ser inicializados.

Exemplo:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

Ao chamar tal função, os parâmetros inicializados podem ser omitidos, os valores default serão usados no lugar deles.

Exemplo:

```
func(123, 0.5);
```

Parâmetros de tipos simples são passados por valor, isto é, modificações da correspondente variável local deste tipo dentro da função chamada não irá se refletir na função chamadora. Arrays de qualquer tipo e dados do tipo estrutura são passados sempre por referência. Se for necessário proibir modificações no conteúdo de um array ou estrutura, os parâmetros destes tipos devem ser declarados com a palavra-chava *const*.

Existe a possibilidade de passar parâmetros de tipos simples por referência. Neste caso, modificações nestes parâmetros de dentro da função chamada afetarão as variáveis correspondentes passadas por referência. A fim de indicar que um parâmetro é passado por referência, coloque o modificador & após do tipo do parâmetro.

Exemplo:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
   for(int i=0; i<OrdersTotal(); i++)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
     }
   x=i;
   y=calculated_tp;
  }
```

Parâmetros passados por referência não podem ser inicializados por valores default.

No máximo 64 parâmetros podem ser passado para um função.

Também Veja

Variáveis de Entrada, Tipo de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Variáveis Estáticas

A classe de armazenamento estático (static) define uma variável estática. O modificador static é indicado antes do tipo do dado.

Exemplo:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

Uma variável estática pode ser inicializada por uma constante ou expressão constante correspondente ao seu tipo, diferentemente de uma variável local simples, que pode ser inicializada por qualquer expressão.

Variáveis estáticas existem a partir do momento de execução do programa e são inicializados somente uma vez antes que a função especializada OnInit() seja chamada. Se valores iniciais não forem especificados, variáveis da classe de armazenamento estático assumem valores iniciais zero.

Variáveis locais declarados com a palavra-chave *static* retém seus valores durante todo o tempo de vida da função. A cada próxima chamada da função, tais variáveis locais contêm os valores que elas tinham durante a chamada anterior.

Quaisquer variáveis em um bloco, exceto parâmetros formais de uma função, podem ser definidas como estáticas. Se uma variável declarada em nível local não for uma variável estática, a memória para tal variável é alocada automaticamente em uma pilha de programa.

Exemplo:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("A função Counter tem sido chamada ",count," tempos");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

Também Veja

© 2000-2024, MetaQuotes Ltd.

Tipos de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criação e Exclusão de Objetos, Membros de Classe Estáticos

© 2000-2024, MetaQuotes Ltd.

Variáveis Globais

Variáveis globais são criadas colocando suas declarações fora de descrições da função. Variáveis globais são definidas no mesmo nível que funções, isto é, elas não são locais de algum bloco.

Exemplo:

```
int GlobalFlag=10;   // Variável global
int OnStart()
  {
   ...
  }
```

O escopo de variáveis globais é o programa inteiro. Variáveis globais são acessíveis a partir de todas as funções definidas no programa. Elas são inicializadas com zero a menos que um outro valor inicial seja explicitamente definido. Uma variável global pode ser inicializada somente por uma constante, ou expressão de constante que corresponde ao tipo dela.

As variáveis globais são inicializadas apenas uma vez após o programa ser carregado na memória do terminal do cliente e antes da primeira manipulação do evento Init. Para as variáveis globais que representam objetos de classe, durante sua inicialização os construtores correspondentes são chamados. Em scripts, as variáveis globais são inicializadas antes de manipular o evento Start.

Observação: Variáveis declaradas em nível global não devem ser confundidas com variáveis globais do terminal cliente que podem ser acessadas usando as funções GlobalVariable...().

Também Veja

Tipos de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Variáveis de Entrada

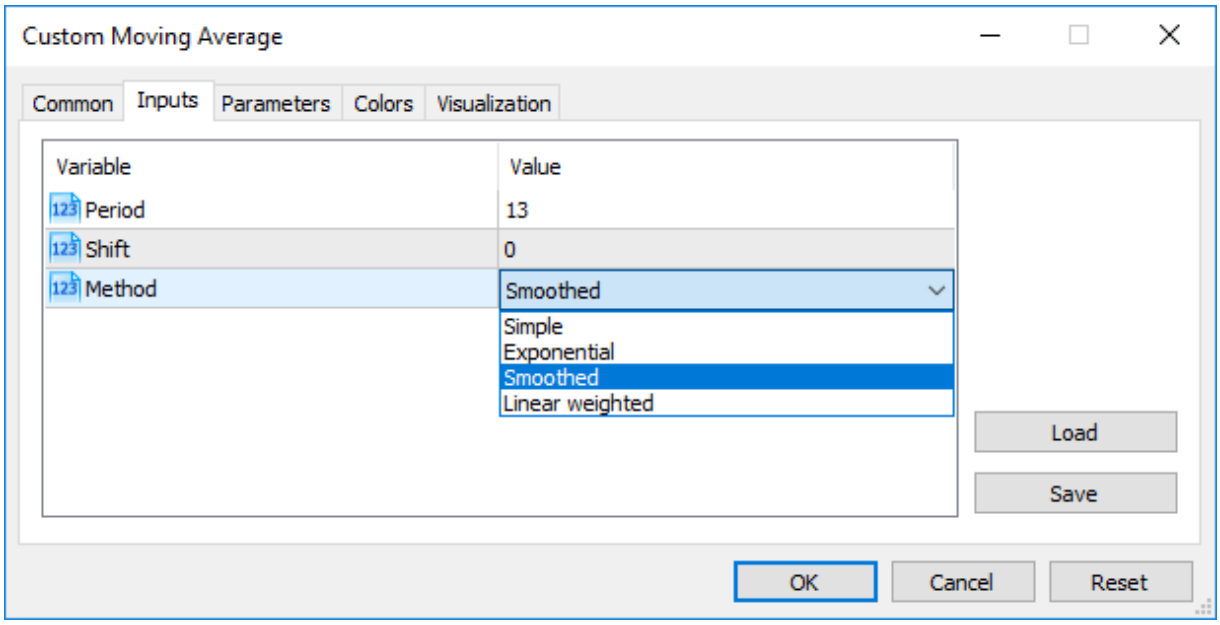
A classe de armazenamento input define uma variável externa. O modificador *input* é indicada antes do tipo de dado. Uma variável com o modificador input não pode ser modificada dentro de programas mql5, tais variáveis podem ser acessadas somente para leitura. Valores de variáveis de entrada podem ser modificados pelo usuário somente a partir da janela de propriedades do programa. Le variabili esterne vengono sempre reinizializzate immediatamente prima che OnInit() venga chiamato.

O comprimento máximo de um nome de variável input é 63 caracteres. Além disso, para um parâmetro de entrada do tipo string o comprimento do valor **máximo** (comprimento da string) pode variar de 191 a 253 caracteres (veja a Observação) O comprimento mínimo é de 0 caracteres (nenhum valor especificado).

Exemplo:

```
//--- parâmetros de entrada
input int            MA_Period=13;
input int            MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

Variáveis de entrada determinam a entrada de parâmetros de um programa. Eles estão disponíveis a partir da janela de Propriedades de um programa.



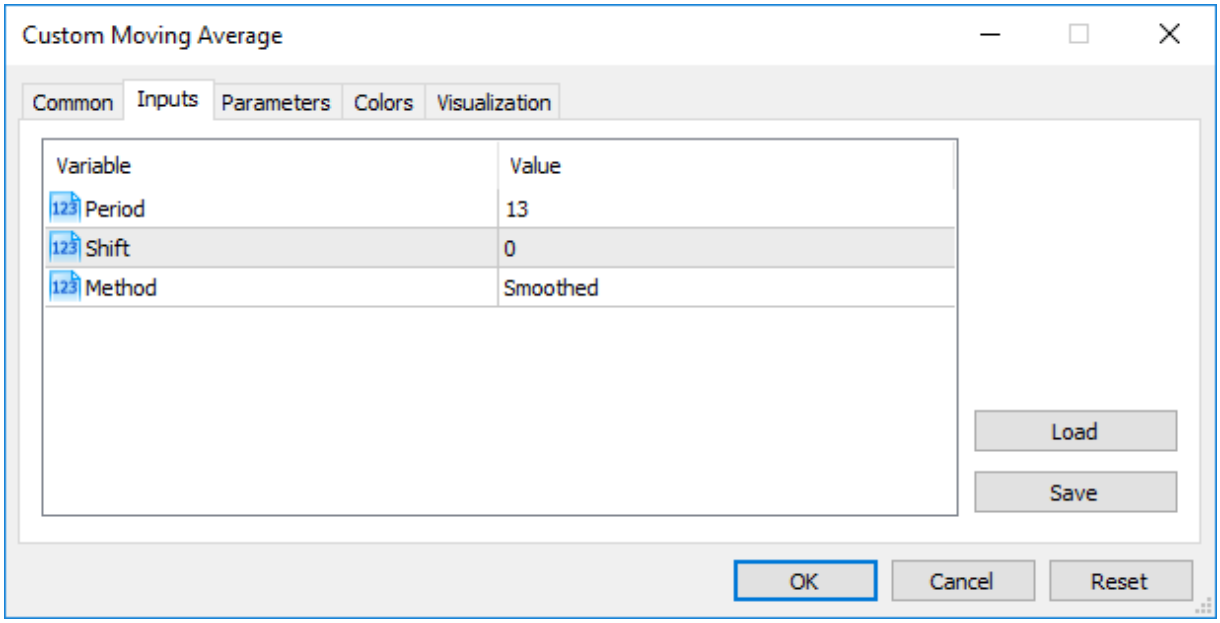
É possível definir uma outra forma de exibir nomes de parâmetros de entrada na aba Parâmetros de Entrada. Para fazer isso, um comentário string é usado, que deve estar localizado após a descrição de um parâmetro de entrada na mesma linha. Assim, nomes mais compreensivos para o usuário podem ser associados aos parâmetros de entrada.

Exemplo:

```
//--- parâmetros de entrada
input int            InpMAPeriod=13;         // Período suavizado
```

© 2000-2024, MetaQuotes Ltd.

```
input int            InpMAShift=0;          // Deslocamento da linha horizontal
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Método de Suavização
```



Observação: Arrays e variáveis de tipos complexos não podem agir como variáveis de entrada.

Observação: O comprimento de um comentário string para as variáveis de entrada não pode exceder 63 caracteres.

Observação: Para variáveis input do tipo string a limitação do comprimento do valor (o comprimento da string) é especificada pelas seguintes condições:

- o valor do parâmetro é representado pela string "parameter_name=parameter_value" (o símbolo '=' é levado em consideração),
- comprimento máximo de representação de 255 caracteres (*total_length_max*=255 ou 254 caracteres excluindo '='),
- comprimento máximo de um nome de parâmetro de string *parameter_name_length* = 63 caracteres.

Assim, o tamanho máximo da string para um parâmetro de string pode ser calculado pela fórmula:

```
parameter_value_length=total_length_max-parameter_name_length=254-parameter_name_lengt
```

Isso dá um tamanho máximo de string de 191 (parameter_name_length=63) a 253 caracteres (parameter_name_length=1).

Passando Parâmetros Ao Chamar Indicadores Customizados a partir de Programas MQL5

Indicadores Customizados são chamados usando a função iCustom(). Após o nome do indicador customizado, os parâmetros devem aparecer em estrita conformidade com a declaração das variáveis de entrada deste indicador customizado. Se os parâmetros indicados são em número menor que as variáveis de entrada declaradas no indicador customizado chamado, os parâmetros faltantes são preenchidos com os valores especificados durante a declaração de variáveis.

© 2000-2024, MetaQuotes Ltd.

Se o indicador customizado usar a função OnCalculate do primeiro tipo (isto é, o indicador é calculado usando o mesmo array dos dados), então um dos valores ENUM_APPLIED_PRICE ou o manuseio de um outro indicador deve ser usado como o último parâmetro ao chamar tal indicador customizado. Todos os parâmetros correspondentes às variáveis de entrada devem ser claramente indicados.

Enumerações como Parâmetros de Entrada

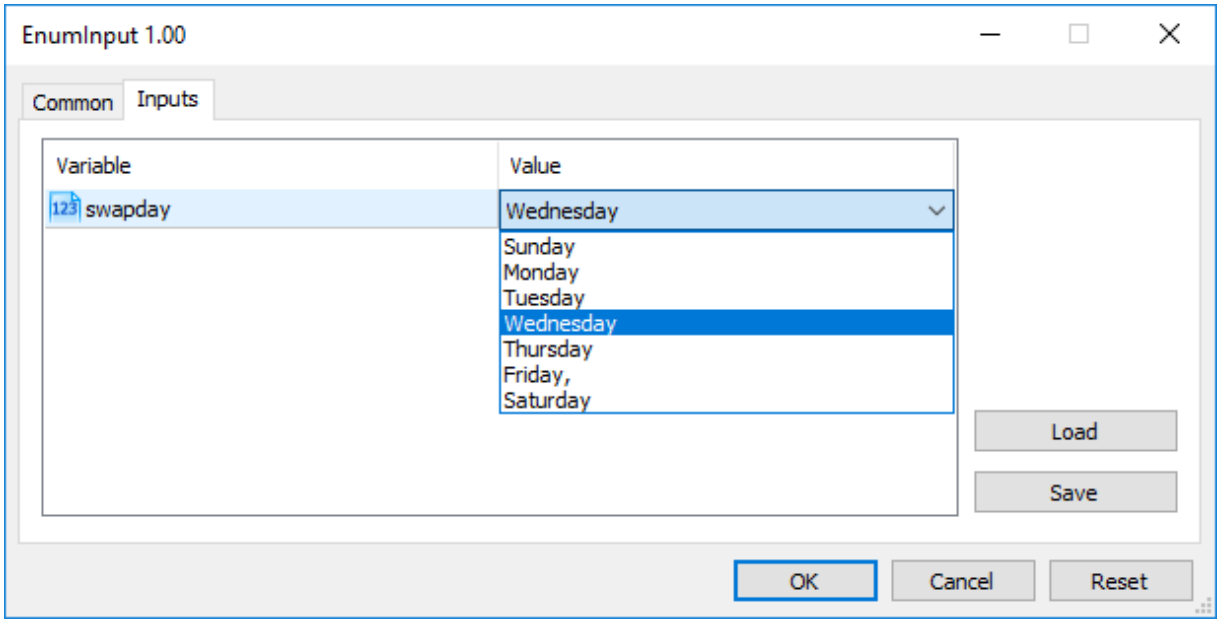
Não apenas enumerações internas fornecidos no MQL5, mas também variáveis definidas pelo usuário podem ser usadas como variáveis de entrada (parâmetros de entrada para programas MQL5). Não apenas enumerações internas fornecidos no MQL5, mas também variáveis definidas pelo usuário podem ser usadas como variáveis de entrada (parâmetros de entrada para programas MQL5).

Exemplo:

```
#property script_show_inputs
//--- dias da semana
enum dayOfWeek
  {
   S=0,      // Domingo
   M=1,      // Segunda-feira
   T=2,      // Terça-feira
   W=3,      // Quarta-feira
   Th=4,     // Quinta-feira
   Fr=5,     // Sexta-feira,
   St=6,     // Sábado
  };
//--- parâmetros de entrada
input dayOfWeek swapday=W;
```

A fim de permitir a um usuário selecionar um valor necessário da janela de propriedades durante o início da execução de um script, nós usamos o comando de pré-processamento #property script show inputs. Iniciamos o script e podemos escolher um dos valores da enumeração dayOfWeek da lista. Iniciamos o script EnumInInput e vamos para a aba Parâmetros de Entrada. Por default, o valor de swapday (dia de cobrança tripla de swap) é Wednesday (W = 3), mas nós podemos especificar qualquer outro valor, e usar este valor para mudar a operação do programa.

© 2000-2024, MetaQuotes Ltd.



O número de possíveis valores de uma enumeração é limitada. A fim de obter o código de erro, a função GetLastError() deve ser chamada. Nomes mnemônicos de membros da enumeração são usados para valores exibidos na lista. Se um comentário é associado com um nome mnemônico, como mostrado no exemplo, o conteúdo do comentário é usado ao invés do nome mnemônico.

Cada valor da enumeração dayOfWeek tem seu valor de 0 a 6, mas na lista de parâmetros, os comentários especificados para cada valor será exibido. Isso fornece flexibilidade adicional para escrever programas com descrições mais claras dos parâmetros de entrada.

Variáveis com Modificador sinput

Variáveis com modificador input não apenas permitem definir parâmetros externos ao iniciar programas mas também são necessárias na otimização de estratégias de negociação no Provador de Estratégia. Cada variável de entrada excluindo-se o do tipo string pode ser usada em optimização.

Algumas vezes, é necessário excluir alguns parâmetros externos do programa da área total de passos do Provador de Estratégia. O modificador de memória sinput foi introduzido para tais casos. sinput significa declaração de variável externa estática (sinput = static input). Significa que a declaração seguinte em um código Expert Advisor

```
sinput       int layers=6;   // Número de camadas
```

será equivalente a declaração completa

```
static input int layers=6;   // Número de camadas
```

A variável declarada com o modificador sinput é um parâmetro de entrada de um programa MQL5. O valor deste parâmetro pode ser modificado ao iniciar o programa. Entretanto, esta variável não pode ser usado na otimização de parâmetros de entrada. Em outras palavras, seu valor não é enumerado ao procurar pelo melhor conjunto ajustado de parâmetros a uma condição específica.

© 2000-2024, MetaQuotes Ltd.

Strategy Tester

Variable	Value	Start	Step	Stop	Steps
Number of layers	6				
Neurons in a layer	**30**	**30**	**1**	**300**	**271**
Number of bars to be analyzed	**13**	**13**	**1**	**130**	**118**
Forecast horizon	**2**	**2**	**1**	**20**	**19**
Network type	0	0	1	10	
					607582

Settings | Inputs | Optimization Results | Agents | Journal

O Expert Advisor mostrado acima tem 5 parâmetros externos. "Número de camadas" é declarado para ser sinput e igual a 6. Este parâmetro não pode ser modificado durante um otimização de estratégia de negociação. Podemos especificar o valor necessário para ele ser usado mais adiante. Os campos Iniciar, Passo, Parar não ficam disponíveis para tal variável.

Portanto, usuários não serão capazes de otimizar este parâmetro após especificarmos o modificador sinput para a variável. Em outras palavras, o usuário do terminal não será capaz de definir os valores inicial e final para ele no Provador de Estratégia para enumeração automática na faixa especificada durante uma otimização.

Porém, existe uma exceção a esta regra: variáveis sinput podem variar em tarefas de optimização usando a função ParameterSetRange(). Esta função foi introduzida especificamente para o programa controlar os conjuntos de valores disponíveis para qualquer variável input, incluindo aquelas declaradas como static input (sinput). A função ParameterGetRange() permite receber os valores variáveis de entrada quando a otimização é iniciada (com o manipulador OnTesterInit()) e redefine uma alteração do valor do passo e um intervalo, no qual uma otimização dos valores dos parâmetros serão enumerados.

Desta forma, ao combinando o modificador sinput com duas funções que trabalham com parâmetros de entrada, isso permite criar regras flexíveis para definir intervalos de otimização de parâmetros de entrada que dependem dos valores de outros parâmetros de entrada.

Agrupando Parâmetros de Entrada

Para facilitar o trabalho com programas MQL5, os parâmetros de entrada podem ser divididos em blocos nomeados usando a palavra-chave group. Isso permite que você separe visualmente alguns parâmetros de outros com base na lógica incorporada neles.

```
input group          "Nome do grupo"
input int            variable1 = ...
input double         variable1 = ...
input double         variable3= ...
```

Todos os parâmetros de entrada após essa declaração são visualmente agrupados no grupo especificado, facilitando para o usuário do programa MQL5 configurar parâmetros ao executar no gráfico ou no testador de estratégia. A indicação de cada grupo é válida até que apareça uma declaração do grupo:

```
input group          "Nome do grupo #1"
input int            group1_var1 = ...
```

© 2000-2024, MetaQuotes Ltd.

```
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Nome do grupo #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Exemplo de EA no qual o bloco de parâmetros de entrada é dividido segundo sua finalidade:

```
input group           "Signal"
input int             ExtBBPeriod    = 20;        // Bollinger Bands period
input double          ExtBBDeviation= 2.0;        // deviation
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;     // BB timeframe

input group           "Trend"
input int             ExtMAPeriod   = 13;         // Moving Average period
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;      // MA timeframe

input group           "ExitRules"
input bool            ExtUseSL       = true;      // use StopLoss
input int             Ext_SL_Points = 50;         // StopLoss in points
input bool            ExtUseTP       = false;     // use TakeProfit
input int             Ext_TP_Points = 100;        // TakeProfit in points
input bool            ExtUseTS       = true;      // use Trailing Stop
input int             Ext_TS_Points = 30;         // Trailing Stop in points

input group           "MoneyManagement"
sinput double         ExtInitialLot = 0.1;        // initial lot value
input bool            ExtUseAutoLot = true;       // automatic lot calculation

input group           "Auxiliary"
sinput int            ExtMagicNumber = 123456;    // EA Magic Number
sinput bool           ExtDebugMessage= true;      // print debug messages
```

Ao iniciar este EA no testador de estratégia, você pode clicar duas vezes no nome do grupo para recolher e expandir o bloco de parâmetros de entrada, além de selecionar todos os seus parâmetros com um clique na caixa de seleção do grupo para otimização.

© 2000-2024, MetaQuotes Ltd.

Strategy Tester

Select expert... | View previous optimization results

Variable	Value	Start	Step	Stop	Steps
Signal					
Bollinger Bands period	20	20	10	60	5
deviation	2	2	0.2	3	6
BB timeframe	15 Minutes	current		30 Minutes	
Trend					
Moving Average period	13	13	1	20	8
MA timeframe	15 Minutes	current		1 Hour	
ExitRules					
use StopLoss	true	false		true	2
StopLoss in points	50	50	10	100	6
use TakeProfit	false	false		true	2
TakeProfit in points	100	100	50	300	5
use Trailing Stop	true	false		true	2
Trailing Stop in points	30	30	10	70	5
MoneyManagement					
initial lot value	0.1				
automatic lot calculation	true	false		true	2
Auxiliary					
EA Magic Number	123456				
print debug messages	true				
					576000

Overview | Settings | Inputs | Backtest | Graph | Optimization Results | Agents | Journal | Start

Veja Também

iCustom, Enumerações, Propriedades de Programas

© 2000-2024, MetaQuotes Ltd.

Variáveis Externas

A palavra-chave extern é usado para declarar identificadores de variáveis como identificadores da classe de armazenamento estática (estático) com tempo de vida global. Estas variáveis existem a partir do inicio do programa e a memória para elas é alocada e inicializada imediatamente após o inicio do programa.

Você pode criar programas que consistem de múltiplos arquivos fontes; neste caso a diretiva para o preprocessador #include é usada. Variáveis declaradas como extern com o mesmo tipo e identificador podem existir em arquivos fontes diferentes de um projeto.

Ao compilar o projeto inteiro, todas as variáveis extern com o mesmo tipo e identificador são associados com uma parte da memória do conjunto (pool) de variáveis globais. Variáveis extern são úteis para compilação separada de arquivos fontes. Variáveis externas podem ser inicializadas, mas apenas uma vez - a existência de várias variáveis externas inicializadas do mesmo tipo e com o mesmo identificador é proibida.

Também Veja

Tipos de Dados, Encapsulamento e Extensibilidade de Tipos, Inicialização de Variáveis, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Inicialização de Variáveis

Qualquer variável pode ser inicializada durante a definição. Se uma variável não é inicializada explicitamente, o valor armazenado nesta variável pode ser qualquer coisa. Inicialização implícita não é usada.

Variáveis globais e estáticas podem ser inicializadas uma vez por uma constante do tipo correspondente ou uma expressão constante. Variáveis locais podem ser inicializados por qualquer expressão, não apenas por uma constante.

A inicialização de variáveis globais e estáticas é realizada apenas uma vez. A inicialização de variáveis locais é feita toda a vez que se chama as funções correspondentes.

Exemplos:

```
int    n       = 1;
string s       = "Olá";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- de tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- inicialização de todos os campos da estrutura com valores zero
MqlTradeRequest request={};
```

A lista de valores dos elementos de um array devem estar entre chaves. Seqüências de inicializações faltantes são consideradas iguais a 0.

Se o tamanho do array inicializado não é especificado, ele é determinado pelo compilador, baseado no tamanho da seqüência de inicialização.

Exemplos:

```
struct str3
  {
   int                low_part;
   int                high_part;
  };
struct str10
  {
   str3               s3;
   double             d1[10];
   int                i3;
  };
void OnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{},100};
   str10 s10_3={{1,0},{1.0}};
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

Para variável de tipo estrutura, inicialização parcial é permitida, bem como para arrays estáticos. Pode-se inicializar um ou mais primeiros elementos de uma estrutura ou array, os demais elementos serão inicializados com zeros neste caso.

Também Veja

Tipos de Dados, Encapsulamento e Tipo de Extensibilidade, Visibilidade Escopo e Tempo de Vida de Variáveis, Criando e Excluindo Objetos

© 2000-2024, MetaQuotes Ltd.

Visibilidade Escopo e Tempo de Vida de Variáveis

Existem dois tipos básicos de escopo: escopo local e escopo global.

Uma variável declarada fora de todas as funções está localizada no escopo global. O acesso a tais variáveis pode ser feito de qualquer lugar no programa. Estas variáveis são localizadas no conjunto (pool) global de memória, assim seus tempos de vida coincidem com o tempo de vida do programa.

Uma variável declarada dentro de um bloco (parte do código entre chaves) pertence ao escopo local. Tal variável não é visível (e portanto não é disponível) fora do bloco no qual está definida. O caso mais comum de declaração local é uma variável declarada dentro de uma função. Uma variável declarada localmente está localizada na pilha (stack), e o tempo de vida de tal variável é igual ao tempo de vida da função.

Já que o escopo de uma variável local é o bloco em que ela está declarada, é possível declarar variáveis com o mesmo nome que variáveis declaradas em outros blocos, mesmo aquelas declaradas em níveis mais acima, até o nível global.

Exemplo:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int        i,limit;
   static int weightsum=0;
   double     sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- define valor vazio para primeiras barras limite
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- calcula primeiro valor visível
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
```

© 2000-2024, MetaQuotes Ltd.

```
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
  }
```

Preste atenção na variável i, declarada na linha

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Seu escopo é somente o ciclo do for; fora deste ciclo existe uma outra variável com o mesmo nome, declarada no começo da função. Além disso, a variável k está declarada no corpo do loop, seu escopo é o corpo do loop.

Variáveis locais podem ser declarados com o especificador de acesso static. Neste caso, o compilador tem uma variável no conjunto (pool) global de memória. Portanto, o tempo de vida de uma variável estática é igual ao tempo de vida do programa. Aqui o escopo de tal variável é limitado ao bloco no qual ela está declarada.

Também Veja

Data Types, Encapsulation and Extensibility of Types, Initialization of Variables, Creating and Deleting Objects

© 2000-2024, MetaQuotes Ltd.

Criação e Exclusão de Objetos

Após um programa MQL5 ser carregado para execução, memória é alocada para cada variável de acordo com seu tipo. De acordo com o nível de acesso, todas as variáveis são dividas em dois tipos - variáveis globais e variáveis locais. De acordo com a classe de memória, elas pode ser parâmetros de entrada de um programa MQL5, estáticas (static) e automáticas. Se necessário, cada variável é inicializada por um valor correspondente. Após ser usada, uma variável é desinicializada e a memória usada por ela é retornada ao sistema executável MQL5.

Inicialização e Desinicialização de Variáveis Globais

Variáveis globais são inicializados automaticamente logo após um programa mql5 ser carregado e antes de qualquer função ser chamada. Durante a inicialização valores iniciais são atribuídos a variáveis de tipos simples e um construtor (se existir algum) é chamado para objetos. Variáveis de entrada são sempre declarados em um nível global, e são inicializados por valores definidos pelo usuário na janela de diálogo durante o começo do programa.

Apesar do fato de que variáveis estáticas serem geralmente declaradas em um nível local, a memória para estas variáveis é pré-alocada, e a inicialização é realizada logo após um programa ser carregado; o mesmo para variáveis globais.

A ordem de inicialização corresponde a ordem de declaração da variável no programa. A desinicialização é realizada em ordem inversa. Este regra é verdadeira apenas para as variáveis que não foram criadas pelo operador new. Tais variáveis são criadas e inicializadas automaticamente logo após carregamento, e são desinicializadas depois do descargamento do programa.

Inicialização e Desinicialização de Variáveis Locais

Se uma variável declarada em um nível local não é estática, memória é alocada automaticamente para tal variável. Variáveis locais, bem como variáveis globais, são inicializados automaticamente no momento que a execução do programa encontra a declaração de uma variável local. Assim, a ordem de inicialização corresponde à ordem de declaração.

Variáveis locais são desinicializados no final do bloco do programa, no qual elas foram declaradas, e na ordem oposta de suas declarações. Um bloco de programa é um operador de composição que pode ser uma parte de um operador de seleção switch, operador de ciclo (for, while, do-while), um corpo de função ou uma parte do operador if-else.

Variáveis locais são inicializados somente no momento em que a execução do programa encontra a declaração da variável. Se durante a execução do programa o bloco, no qual a variável está declarada, não for executada, tal variável não é inicializada.

Inicialização e Desinicialização de Objetos Colocados

Um caso especial é aquele com ponteiros de objeto, porque a declaração de um ponteiro não implica em inicialização de objetos correspondentes. Objetos colocados dinamicamente são inicializados somente no momento em que a instância da classe é criada pelo operador new. Inicialização de objetos pressupõe chamada de um construtor da classe correspondente. Se não existir construtor correspondente na classe, seus membros de tipo simples não serão automaticamente inicializados; membros de tipos string, array dinâmico e objeto complexo serão automaticamente inicializados.

© 2000-2024, MetaQuotes Ltd.

Ponteiros podem ser declarados em nível local ou global; e eles podem ser inicializados pelo valor vazio de NULL ou pelo valor do ponteiro do mesmo ou herdado tipo. Se o operador *new* é chamado para um ponteiro declarado em um nível local, o operador *delete* para este ponteiro deve ser chamado antes de sair do nível. Caso contrário, o ponteiro estará perdido e a exclusão explícita do objeto falhará.

Todos os objetos criados pela expressão *object_pointer*=new *Class_name*, devem ser então excluídos pelo operador delete(*object_pointer*). Se por algum razão tal variável não for excluída por meio do operador delete quando o programa estiver concluído, a correspondente entrada aparecerá no diário "Experts". Pode-se declarar várias variáveis e atribuir um ponteiro de um objeto para todas elas.

Se um objeto criado dinamicamente tiver um construtor, este construtor será chamado no momento da execução do operador *new*. Se um objeto tiver um destrutor, ele será chamado durante a execução do operador *delete*.

Assim objetos colocados dinamicamente são criados somente no momento em que são criados pelo operador *new*, e são certamente excluídos tanto pelo operador *delete* ou automaticamente pelo sistema de execução do MQL5 durante a descarga do programa. A ordem de declaração de ponteiros criados dinamicamente não influenciam na ordem de suas inicializações. A ordem de inicialização e desinicialização é totalmente controlada pelo programador.

Características Transitórias de Variáveis

A principal informação sobre a ordem de criação, exclusão, sobre chamadas de construtores e destrutores é dada na tabela a baixo.

	Variável automática global	Variável automática local	Objeto criado dinamicamente
Inicialização	logo após um programa MQL5 ser carregado	quando a linha de código onde ela está declarada é alcançada durante execução	na execução do operador **new**
Ordem de inicialização	na ordem da declaração	na ordem da declaração	independente da ordem de declaração
Desinicialização	antes de um programa MQL5 ser descarregado	quando a execução sai do bloco da declaração	quando o operador **delete** é executado ou antes de um programa mql5 ser descarregado
Ordem de desinicialização	na ordem oposta da ordem de inicialização	na ordem oposta da ordem de inicialização	independente da ordem de inicialização
Chamada de construtor	no carregamento do programa MQL5	na inicialização	na execução do operador *new*
Chamada de destrutor	na descarga do programa MQL5	quando sair do bloco onde a variável foi inicializada	na execução do operador *delete*
Logs de Erro	mensagem de log no diário "Experts" sobre	mensagem de log no diário "Experts" sobre	mensagem de logo no diário "Experts" sobre

© 2000-2024, MetaQuotes Ltd.

	Variável automática global	Variável automática local	Objeto criado dinamicamente
	a tentativa de excluir um objeto criado automaticamente	a tentativa de excluir um objeto criado automaticamente	objetos criados dinamicamente não excluídos na descarga de um programa mql5

Também Veja

Tipos de Dados, Encapsulamento e Tipos de Extensibilidade, Inicialização de Variáveis, Visibilidade Escope e Tempo de Vida de Variáveis

© 2000-2024, MetaQuotes Ltd.

Preprocessador

Preprocessador é um subsistema especial do compilador MQL5 que é destinado à preparação do código fonte do programa imediatamente antes do programa ser compilado.

O preprocessador permite melhorar a legibilidade do código fonte. O código pode ser estruturado por meio de inclusão de arquivos específicos contendo códigos fonte de programas-mql5. A possibilidade de atribuir nomes mnemônicos a constantes específicas contribui para melhoria da legibilidade do código.

O preprocessador também permite determinar parâmetros específicos de programas MQL5:

- Declarar constantes
- Definir propriedades de programa
- Incluir arquivos no texto do programa
- Importar funções
- Conditional compilation

As diretivas do pré-processador são usadas pelo do compilador para pré-processamento do código fonte antes de sua compilação. A diretiva sempre começa com o # (jogo da velha), por isso o compilador proíbe o uso deste símbolo nos nomes das variáveis, funções, etc.

Cada diretiva é descrita por um registro separado e opera antes da mudança de linha. Não se podem utilizar várias diretivas num só registro. Se o registro da diretiva for muito grande, ele pode ser dividido em várias linhas usando a barra invertida '\', neste caso, a próxima linha é considerada uma continuação do registro da diretiva.

```
//+------------------------------------------------------------------+
//|  pseudo-operador foreach                                         |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- rastreamento da matriz usando o ForEach
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* Resultado de saída
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
```

© 2000-2024, MetaQuotes Ltd.

```
   3: array[3]=45
*/
```

Para o compilador, estas três linhas da diretiva #define aparecerão como uma linha longa. Neste exemplo, também é usado o caractere duplo do jogo da velha ##, ele é chamado operador de fusão e é usado nas macros #define, para combinar dois tokens numa macro. O operador de fusão de tokens não pode ser o primeiro ou o último na definição de macro.

© 2000-2024, MetaQuotes Ltd.

Substituição de Macro (#define, #undef)

As diretivas do pré-processador são usadas pelo do compilador para pré-processamento do código fonte antes de sua compilação. A diretiva sempre começa com o # (jogo da velha), por isso o compilador proíbe o uso deste símbolo nos nomes das variáveis, funções, etc.

Cada diretiva é descrita por um registro separado e opera antes da mudança de linha. Não se podem utilizar várias diretivas num só registro. Se o registro da diretiva for muito grande, ele pode ser dividido em várias linhas usando a barra invertida '\', neste caso, a próxima linha é considerada uma continuação do registro da diretiva.

A diretiva #define pode ser usada para atribuir nomes mnemônicos a constantes. Existem duas formas:

```
#define identificador expressão                 // forma sem parâmetro
#define identificador(par1,... par8) expressão  // forma paramétrica
```

A diretiva #define substitui em **expressão** todas as entradas encontradas adiante do *identificador* no texto fonte. O *identificador* é substituído somente se ele é um símbolo (token) separado. O *identificador* não é substituído se ele é parte de um comentário, parte de uma string, ou parte de um outro identificador mais longo.

O identificador de constante é governado pelas mesmas regras de nomes de variáveis. O valor pode ser de qualquer tipo:

```
#define ABC               100
#define PI                3.14
#define COMPANY_NAME      "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

Uma **expressão** pode consistir de vários símbolos (tokens), tais como palavras chave, constantes, expressões constantes e não-constantes. Uma **expressão** termina com o fim da linha e não pode ser transferida para a próxima linha.

Exemplo:

```
#define TWO        2
#define THREE      3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
// Resultado
// 2 + 3*2 = 8
```

© 2000-2024, MetaQuotes Ltd.

```
// (2 + 3)*2 = 10
```

Forma paramétrica do #define

Na forma paramétrica, todas as subseqüentes entradas encontradas do identificador serão substituídas pela expressão levando em conta os respectivos parâmetros. Por exemplo:

```
 // exemplo com dois parâmetros a e b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
expressão double c=MUL(A,B);
é equivalente a double c=((2+3)*(5-1));
*/
// Resultado
// c=20
```

Se assegure de colocar os parâmetros entre parênteses ao usar os parâmetros na expressão, uma vez que isso ajudará a evitar erros não óbvios que são difíceis de encontrar. Se reescrevermos o código sem o uso de parêntesis, o resultado será diferente:

```
 // exemplo com dois parâmetros a e b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
expressão double c=MUL(A,B);
é equivalente a double c=2+3*5-1;
*/
// Resultado
// c=16
```

Ao usar a forma paramétrica, são permitidos no máximo 8 parâmetros.

```
 // forma paramétrica correta
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // um parâmetro - 'text'

 // forma paramétrica incorreta
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // mais do que 8 p
```

© 2000-2024, MetaQuotes Ltd.

The #undef directive

The #undef directive cancels declaration of the macro substitution, defined before.

Example:

```
#define MACRO

void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MACRO is not defined in ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Resultado:
 MACRO is defined in func1
 MACRO is not defined in func2
*/
```

Também Veja

Identificador, Constantes de Caractere

© 2000-2024, MetaQuotes Ltd.

Propriedades de Programa (#property)

Todo programa mql5 permite especificar parâmetros específicos adicionais denominados #property que ajudam o terminal de cliente a prover adequadamente serviços a programas sem a necessidade de iniciá-los explicitamente. Isso diz respeito a configurações externas de indicadores, em primeiro lugar. Propriedades descritas em arquivos incluídos (included files) são completamente ignoradas. Propriedades devem ser especificadas no arquivo MQL5 principal.

```
#property identifier value
```

O compilador escreverá valores declarados na configuração do módulo executado.

Constante	Tipo	Descrição
icon	string	Caminho para o arquivo da imagem que será usada como um ícone do programa EX5. Regras de especificação de caminho são as mesmas para recursos. A propriedade deve ser especificada no módulo principal com o código fonte MQL5. O arquivo de ícone deve estar no formato ICO.
link	string	Link para o website da empresa
copyright	string	O nome da empresa
version	string	Versão do programa, máximo de 31 caracteres
description	string	Breve descrição em texto do programa MQL5. Pode haver vários *description*, cada um descrevendo uma linha do texto. O comprimento total de todo *description* não pode exceder 511 caracteres incluindo a quebra de linha.
stacksize	int	Tamanho do stack no programa MQL5. É necessário um stack de tamanho satisfatório ao executar chamadas recursivas da função. Ao iniciar um script ou um Expert Advisor no gráfico, é alocado um stack de pelo menos 8 MB. No caso de indicadores, o tamanho da pilha é sempre fixo e igual a 1 MB. Quando um programa é iniciado no testador de estratégia, para ele é sempre alocado um stack de 16 MB.
library		Uma biblioteca (library); nenhuma função start é atribuída, funções com o modificador export podem ser importadas em outros programas mql5
indicator_applied_price	int	Especifica o valor padrão para o campo "Aplicar a". Pode-se especificar um dos valores de ENUM_APPLIED_PRICE. Se a propriedade não é especificada, o valor padrão é PRICE_CLOSE
indicator_chart_window		Exibe o indicador na janela do gráfico

© 2000-2024, MetaQuotes Ltd.

Constante	Tipo	Descrição
indicator_separate_window		Exibe o indicador em uma janela separada
indicator_height	int	Altura (height) fixa da sub-janela do indicador em pixels (property INDICATOR_HEIGHT)
indicator_buffers	int	Número de buffers para cálculo do indicador
indicator_plots	int	Número de séries gráficas no indicador
indicator_minimum	double	O limite de escala inferior para uma janela de indicador separada
indicator_maximum	double	O limite de escala superior para uma janela de indicador separada
indicator_labelN	string	Define o rótulo para a N-ésima série gráfica exibida na Janela de Dados. Para séries gráficas que requerem múltiplos buffers de indicador (DRAW_CANDLES, DRAW_FILLING e outros), os nomes de rótulos são definidos por meio do separador ';'.
indicator_colorN	color	A cor de exibição da linha N, onde N é o número da série gráfica; numeração começando em 1
indicator_widthN	int	Espessura de linha na série gráfica, onde N - número da série gráfica, numeração começando em 1
indicator_styleN	int	Estilo da linha na série gráfica, especificado a partir dos valores de ENUM_LINE_STYLE. N - número de séries gráficas, numeração começando em 1
indicator_typeN	int	Tipo de plotagem gráfica, especificado a partir de valores de ENUM_DRAW_TYPE. N - número de séries gráficas, numeração começando em 1
indicator_levelN	double	Nível horizontal de N em uma janela de indicador separada
indicator_levelcolor	color	Cor dos níveis horizontais do indicador
indicator_levelwidth	int	Espessura dos níveis horizontais do indicador
indicator_levelstyle	int	Estilo dos níveis horizontais do indicador
script_show_confirm		Exibe uma janela de confirmação antes de executar o script
script_show_inputs		Exibe uma janela com as propriedades antes de executar o script e desabilita esta janela de confirmação

© 2000-2024, MetaQuotes Ltd.

Constante	Tipo	Descrição
tester_indicator	string	Nome de um indicador customizado no formato de *"nome_indicador.ex5"*. Indicadores que requeiram teste são definidos automaticamente a partir da função iCustom(), se o correspondente parâmetro é definido através de uma string constante. Para todos os demais casos (uso da função IndicatorCreate() ou uso de uma string não-constante no parâmetro que define o nome do indicador) esta propriedade é requerida
tester_file	string	Nome de arquivo para um testador com a indicação de extensão, entre aspas duplas (como uma string constante). O arquivo especificado será passado para o Testador de Estratégias. Arquivos de entrada para serem testados, se forem necessários, devem sempre ser especificados.
tester_library	string	Nome da biblioteca (library) com a extensão, entre aspas duplas. Uma biblioteca pode ter extensão dll ou ex5. Bibliotecas que requeiram teste são definidas automaticamente. Contudo, se alguma biblioteca for usada por um indicador customizado, esta propriedade é requerida
tester_set	string	Nome de arquivo set com valores e o step dos parâmetros de entrada. O arquivo especificado será transferido para testador antes do teste ou otimização. O nome do arquivo deve ser especificado com a extensão e usando aspas como uma string constante. Se, no nome do arquivo set, você especificar o nome do expert e o número de versão "<expert_name>_<number>.set", ele será automaticamente adicionado ao menu para carregar as versões dos parâmetros de acordo com o número de versão <number>. Por exemplo, o nome "MACD Sample_4.set" indica que se trata de um arquivo set para o expert "MACD Sample.mq5" com um número de versão igual 4. Para estudar formato é recomendado salvar manualmente as configurações de teste/otimização no testador de estratégias, e, em seguida, abrir o arquivo set criado dessa maneira.
tester_no_cache	string	O testador de estratégia, durante a otimização, salva todos os resultados das corridas no cache de otimização em que para cada conjunto de parâmetros de entrada é salvo o resultado do teste.

© 2000-2024, MetaQuotes Ltd.

Constante	Tipo	Descrição
		Isso permite, durante uma nova otimização, nos mesmos parâmetros obter os resultados prontos sem recálculo e demora. Mas para algumas tarefas - por exemplo, em cálculos matemáticos - pode ser necessário realizar cálculos, independentemente da disponibilidade de resultados prontos no cache de otimização. Nesse caso, o arquivo deve incluir a propriedade *tester_no_cache*. Além disso, embora isso, os próprios resultados do teste serão armazenados no cache, para que você possa ver todos os dados nas corridas concluídas no testador de estratégias.
tester_everytick_calculate	string	No testador de estratégias, os indicadores são calculados somente quando são acessados para obter dados, isto é, somente no momento em que são solicitados os valores dos buffers de indicador. Isto dá uma aceleração significativa durante o teste e a otimização, se você não precisar obter os valores do indicador em cada tick. A propriedade *tester_everytick_calculate* permite - durante o teste - ativar forçosamente o modo de cálculo do indicador em cada tick. Os indicadores no testador de estratégia também são contados forçosamente em cada tick nos seguintes casos: • ao testar em modo visual; • se no indicador houver as funções EventChartCustom, OnChartEvent, OnTimer; • se o indicador for criado pelo compilador com número de compilação abaixo de 1916. Esta propriedade se aplica apenas ao trabalho no testador de estratégias, no terminal os indicadores são sempre contados em cada tick entrante.
optimization_chart_mode	string	Indica o tipo de gráfico e os nomes de dois parâmetros de entrada que serão usados para visualizar resultados de otimização. Por exemplo, “3d, InpX, InpY” significa que um gráfico 3D com eixos de coordenadas será mostrado com base nos valores dos parâmetros InpX e InpY iterados. Desse modo, com a ajuda desta propriedade, você pode especificar diretamente no código os parâmetros

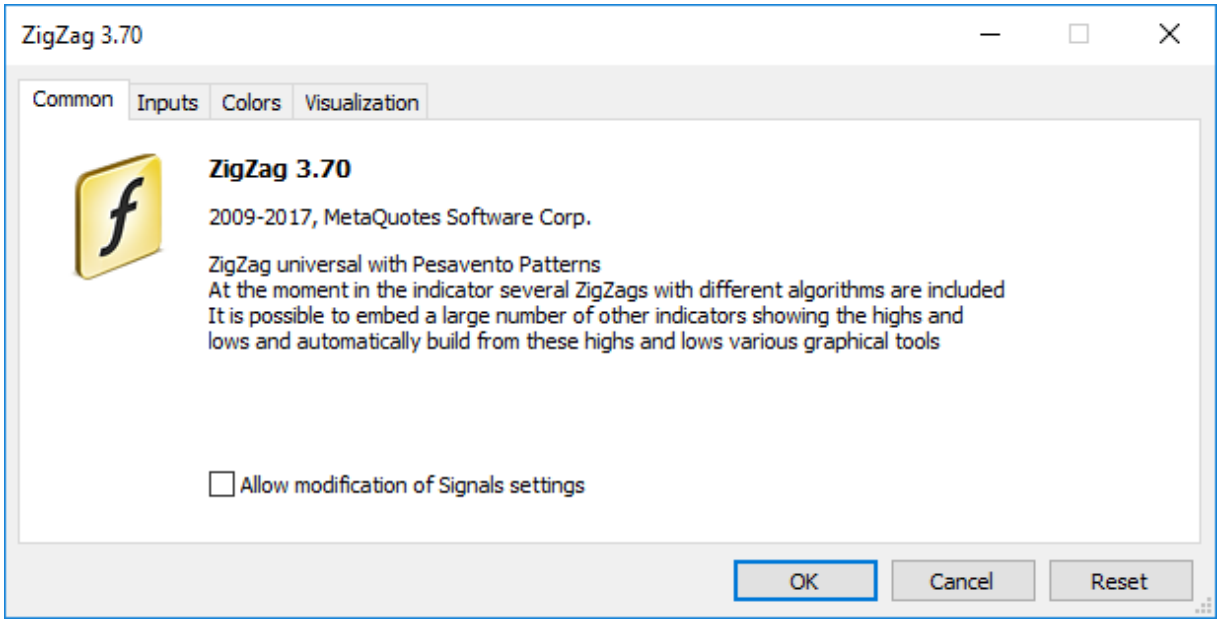
© 2000-2024, MetaQuotes Ltd.

Constante	Tipo	Descrição
		que serão usados para exibir o gráfico de otimização e o tipo do gráfico em si. Possíveis variantes: • "3d, input_parameter_name1, input_parameter_name2" - gráfico de visualização 3D que pode ser girado, ampliado, afastado. É construído de acordo com dois parâmetros de entrada. • "2d, input_parameter_name1, input_parameter_name2" - gráfico plano na forma de grade, cujas células são sombreadas em cores, dependendo do resultado. É construído de acordo com dois parâmetros de entrada. • "1d, input_parameter_name1, input_parameter_name2" - gráfico linear em que os resultados são classificados pelo parâmetro especificado. Cada corrida é exibida na forma de um ponto, é construída de acordo com o parâmetro de entrada. • "0d, input_parameter_name1, input_parameter_name2" - gráfico regular com resultados classificados de acordo com a ordem de chegada do resultado da corrida. Cada corrida é exibida como um ponto. Não requer especificação de parâmetros, mas esses parâmetros serão usados ao alternar manualmente para outro tipo de gráfico. Você pode especificar apenas o tipo de gráfico e não especificar um ou dois parâmetros de entrada, pois, nesse caso, o próprio terminal selecionará os parâmetros necessários para exibir o gráfico de otimização.

Exemplo de Descrição e Número de Versão

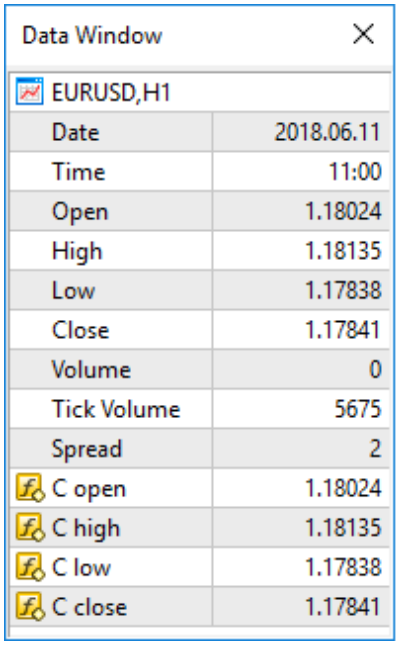
```
#property version     "3.70"      // Versão atual do Expert Advisor
#property description "ZigZag universal com Pesavento Patterns"
#property description "Atualmente estão inclusos no indicador vários ZigZags com difer
#property description "É possível embutir um grande número de outros indicadores exibi
#property description "fundos e automaticamente construir a partir destes topos e func
```

© 2000-2024, MetaQuotes Ltd.



Exemplos especificando um rótulo separado para cada buffer do indicador ("C open; C high; C low; C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```



© 2000-2024, MetaQuotes Ltd.

Incluindo Arquivos (#include)

A linha de comando *#include* pode ser colocada em qualquer lugar no programa, mas geralmente todas as inclusões são colocadas no começo do código fonte. Formato da chamada:

```
#include <file_name>
#include "file_name"
```

Exemplos:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

O preprocessador substitui a linha *#include <file_name>* com o conteúdo do arquivo WinUser32.mqh. Colchetes angulares ("<>") indicam que o arquivo WinUser32.mqh file será obtido a partir do diretório padrão (geralmente *terminal_installation_directory\MQL5\Include*). O diretório corrente não é visível.

Se o nome do arquivo é colocado entre aspas, a procura é feita no diretório corrente (que contém o arquivo fonte principal). O diretório padrão não é visível.

Também Veja

Standard Library, Importando Funções

© 2000-2024, MetaQuotes Ltd.

Importando Funções (#import)

Funções são importadas a partir de módulos MQL5 compilados (arquivos *.ex5) e a partir de módulos do sistema operacional (arquivos *.dll). O nome do módulo é especificado na diretiva *#import*. Para que o compilador formate corretamente a chamada da função importada e organize a correta transferência de parâmetros, é necessária uma descrição completa das funções. Descrições de função sucedem imediatamente à diretiva *#import "nome do módulo"*. O novo comando *#import* (pode-se sem parâmetros) completa o bloco da descrição das funções importadas.

```
#import "file_name"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Funções importadas podem ter qualquer nome. Funções importadas podem ter qualquer nome. Você pode importar simultaneamente módulos - de diferentes funções - com o mesmo nome. Funções importadas podem ter nomes que coincidem com os nomes de funções internas. A operação de resolução de contexto determina quais das funções devem ser chamadas.

A ordem de pesquisa do arquivo especificado após a palavra-chave #import é descrita na seção Chamada de Funções Importadas.

Já que funções importadas estão do lado de fora do módulo compilado, o compilador não pode verificar a validade dos parâmetros passados. Portanto, para evitar erros em tempo de execução, deve-se descrever com precisão a composição e ordem dos parâmetros passados para funções importadas. Parâmetros passados para funções importadas (tanto de EX5, como de módulo DLL) podem ter valores por padrão.

Nas funções importadas, você não pode usar como parâmetros:

- pointers (*);
- links de objetos que contenham arrays dinâmicos e/ou ponteiros.

Nas funções importadas da DLL, você não pode passar - como parâmetros - classes, arrays de string ou objetos complexos contendo strings e/ou arrays dinâmicos de qualquer tipo.

Exemplos:

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
```

© 2000-2024, MetaQuotes Ltd.

```
#import
```

Para importar funções durante a execução de um programa mql5, é usada vinculação inicial (early binding). Isso significa que a biblioteca é carregada durante a carga de um programa usando seu programa ex5.

Não é recomendado que usar o nome completo do módulo a carregar, como *Drive:\Directory\FileName.Ext*. Bibiliotecas MQ5 são carregadas a partir da pasta *terminal_dir\MQL5\Libraries*.

Se a função importada tiver diferentes opções de chamada para as versões de 32 e 64 bits do Windows, você deverá importar ambas e chamar explicitamente a variante de função desejada com ajuda da variável _IsX64.

Exemplo:

```
#import "user32.dll"
//--- para um sistema de 32 bits
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- para um sistema de 64 bits
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit usa a variação correta do  MessageBoxW()   |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   // --- se tivermos o Windows de 64 bits
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"Exemplo de como chamar a versão de 64 bits do MessageBoxW"
     }
   else  // temos o Windows de 32 bits
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"Exemplo de chamada de uma versão de 32 bits do MessageBoxW
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

© 2000-2024, MetaQuotes Ltd.

Importando funções de bibliotecas .NET

Para trabalhar com as funções da biblioteca .NET, basta importar a própria DLL sem especificar funções específicas. O MetaEditor importa automaticamente todas as funções com as quais é possível o trabalho:

- Estruturas simples (POD, plain old data), isto é, estruturas que contêm apenas tipos de dados simples.
- Funções estáticas públicas cujos parâmetros usam apenas tipos simples e estruturas de POD ou seus arrays.

Para chamar funções da biblioteca, basta importá-la:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

O código C# para a função Inc da classe TestClass é semelhante ao seguinte:

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

Como resultado, o script retornará o valor 42.

Veja Também

Incluindo Arquivos

© 2000-2024, MetaQuotes Ltd.

Conditional Compilation (#ifdef, #ifndef, #else, #endif)

As diretivas do pré-processador são usadas pelo do compilador para pré-processamento do código fonte antes de sua compilação. A diretiva sempre começa com o # (jogo da velha), por isso o compilador proíbe o uso deste símbolo nos nomes das variáveis, funções, etc.

Cada diretiva é descrita por um registro separado e opera antes da mudança de linha. Não se podem utilizar várias diretivas num só registro. Se o registro da diretiva for muito grande, ele pode ser dividido em várias linhas usando a barra invertida '\', neste caso, a próxima linha é considerada uma continuação do registro da diretiva.

As diretrizes de compilação condicional do pré-processador permitem compilar ou pular uma parte do programa dependendo do cumprimento de uma determinada condição.

Essa condição pode assumir uma das seguintes formas.

```
#ifdef identifier
   // O código aqui é compilado se o identificador já tiver sido definido para o pré-p
#endif
```

```
#ifndef identifier
   // O código localizado aqui é compilado se o identificador não está definido atualm
#endif
```

Qualquer uma das diretrizes de compilação condicional pode ser seguida por qualquer número de linhas possivelmente contendo a diretiva #else e terminando com #endif. Se a condição verificada for verdadeira, as linhas entre #else e #endif serão ignoradas. Se a condição verificada não for cumprida, todas as linhas entre a verificação e a diretiva #else (ou a diretiva #endif se a primeira estiver ausente) serão ignoradas.

Exemplo:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Test mode");
   #else
      Print("Normal mode");
   #endif
  }
```

Dependendo do tipo de programa e do modo de compilação, os padrões macros são definidos da seguinte forma:

__MQL5__ macro é definido ao compilar o arquivo *.mq5, __MQL4__ macro é definido ao compilar o *.mq4.

© 2000-2024, MetaQuotes Ltd.

_DEBUG macro é definido ao compilar o modo de depuração.
_RELEASE macro É definido ao compilar no modo de liberação.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Olá do compilador MQL5 [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Olá do compilador MQL5 [RELEASE]");
        #endif
     #endif
   #else
      #ifdef __MQL4__
         #ifdef _DEBUG
           Print("Olá do compilador MQL4 [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Olá do compilador MQL4 [RELEASE]");
           #endif
        #endif
     #endif
   #endif
  }
```

© 2000-2024, MetaQuotes Ltd.

Programação Orientada a Objetos

Programação orientada a objetos (OOP) é programar basicamente focada em dados, enquanto dados e comportamento estão inseparavelmente ligados. Juntos, dados e comportamento, constituem uma classe, enquanto objetos são instâncias de classe.

Os componentes da abordagem orientada a objetos são:

- Encapsulamento e extensibilidade de tipo
- Herança
- Polimorfismo
- Sobrecarga
- Funções virtuais

OOP considera computação como modelagem de comportamento. O item modelado é o objeto representado pelas abstrações computacionais. Suponha que nós quiséssemos escrever "Tetris", um jogo bem conhecido. Para fazer isso, nós devemos aprender como modelar a aparência de uma forma aleatórias compostas de quatro quadrados ligados nas bordas. Também nós precisaríamos regular a velocidade de queda das figuras e definir operações de rotação e deslocamento. A movimentação das formas na tela é limitada pelas bordas da peça, este requerimento também deve ser modelado. Além disso, linhas preenchidas de cubos devem ser destruídas e pontos conseguidos devem ser contados.

Assim, este jogo de fácil entendimento requer a criação de vários modelos - modelo de forma, modelo de peça, modelo de movimentação da figura, e assim por diante. Todas estes modelos são abstrações, representadas por cálculos no computador. Para descrever estes modelos, o conceito de Tipo de Dados Abstratos, ADT (tipo de dados complexos), é usado. Estritamente falando, o modelo do movimento das "formas" no jogo não é um tipo de dado, mas é um conjunto de operações sobre o tipo de dado "forma", usando as restrições do tipo de dado "peça".

Objetos são variáveis de classe. A programação orientada a objetos permite a você facilmente criar e usar ADT. A programação orientada a objetos usa o mecanismo de herança. O benefício da herança está no fato de permitir obter tipos derivados a partir de tipos de dados já definidos por um usuário.

Por exemplo, para criar formas Tetris, é conveniente primeiro criar uma classe base Forma. As outras classes representando todos os sete possíveis tipos de forma podem ser derivadas desta base. O comportamento das figuras é definido na classe base, enquanto que a implementação do comportamento de cada figura separada é definida em classes derivadas.

Em OOP, objetos são responsáveis por seu comportamento. O desenvolvedor de ADT deve incluir um código para descrever qualquer comportamento que normalmente seria esperado dos objetos correspondentes. O fato do objeto ser responsável por seu comportamento, simplifica enormemente a tarefa de programação deste objeto pelo usuário.

Se quiséssemos desenhar uma figura na tela, precisaríamos conhecer onde o centro desta figura estaria e como desenhá-la. Se um forma separada sabe como se desenhar, o programador deve enviar uma mensagem de "desenhe" ao usar tal forma.

A linguagem MQL5 é semelhante ao C++, e ela também tem mecanismos de encapsulamento para a implementação de ADT. Por um lado, o encapsulamento esconde os detalhes internos da implementação de um tipo particular, e por outro lado, torna acessível externamente funções que podem influenciar objetos deste tipo. Detalhes de implementação podem ser inacessíveis a um programa que usa este tipo.

© 2000-2024, MetaQuotes Ltd.

O conceito de OOP tem um conjunto de conceitos relacionados, incluindo o seguinte:

- Simulação de ações do mundo real
- Tipos de dados definidos pelo usuário
- Ocultamente de detalhes de implementação
- Possibilidade de reutilização de código através de herança
- Interpretação de chamadas de função durante execução

Alguns destes conceitos são bastante vagos, alguns são abstratos, outros são gerais.

© 2000-2024, MetaQuotes Ltd.

Encapsulamento e Extensibilidade de Tipo

OOP é uma abordagem balanceada para escrever software. Dados e comportamento são empacotados juntos. Este encapsulamento cria tipos de dados definidos pelo usuário, estendendo os tipos de dados da linguagem e interagindo com eles. Extensibilidade de tipos é uma oportunidade de adicionar à linguagem tipos de dados definidos pelo usuário, que são também fáceis de usar, tanto quanto os tipos básicos.

Um tipo de dado abstrato (ADT), por exemplo, uma string, é uma descrição do ideal, bem conhecido tipo de comportamento.

O usuário de string sabe que as operações de string, tais como concatenação ou impressão, têm um certo comportamento. Operações de concatenação e impressão são chamados métodos.

Uma certa implementação de ADT pode ter algumas restrições, por exemplo, strings podem ser limitados no comprimento. Estas limitações afetam o comportamento aberto a todos. Ao mesmo tempo, detalhes da implementação privada ou interna não afetam diretamente a jeito como o usuário vê o objeto. Por exemplo, a string é freqüentemente implementado como um array, enquanto o endereço de base interno deste array e seu nome não são essenciais para o usuário.

Encapsulamento é a habilidade de ocultar os detalhes de implementação quando a interface aberta para o tipo definido pelo usuário é fornecida. Em MQL5, assim como em in C++, definições de classe e estrutura (class e struct) são usadas para as provisões de encapsulamento em combinação com as palavras-chave de acesso private, protected e public.

A palavra-chave public mostra que o acesso aos membros que ficam depois dela, é aberto sem restrições. Sem esta palavra-chave, membros de classe ficariam bloqueados por default. Membros privados são acessíveis somente por funções membro de sua própria classe.

Funções de classe protegidas são disponíveis para funções de classe não apenas em sua classe, mas também em suas classes herdeiras. Funções de classe publicas são disponíveis para qualquer função dentro do escopo da declaração da classe. A proteção torna possível ocultar parte da implementação da classe, evitando assim mudanças inesperadas na estrutura dos dados. Restrição de acesso ou ocultamento de dados é uma característica da programação orientada a objetos.

Geralmente, funções de classe são protegidas e declaradas com o modificador protected, a leitura e a escrita de valores são realizadas por meio do uso de métodos especiais chamados set e get que são definidos por meio do modificador de acesso public.

Exemplo:

```
class CPerson
  {
protected:
   string            first_name;                 // nome
public:
   void              SetName(string n){m_name=n;}// atribui o nome
   string            GetName(){return (m_name);} // retorna o nome
  };
```

© 2000-2024, MetaQuotes Ltd.

Esta abordagem oferece várias vantagens. Primeiro, por meio do nome da função nós podemos entender o que ela faz - define (sets) ou obtém (gets) o valor de um membro de classe. Segundo, talvez no futuro precisemos modificar o tipo da variável m_name na classe CPerson ou em qualquer de suas classes derivadas.

Neste caso, necessitaríamos apenas alterar a implementação das funções SetName() e GetName(), enquanto que objetos da classe CPerson estariam disponíveis para uso em um programa sem nenhuma alteração de código, porque o usuário nem ao menos saberia que o tipo de dado do m_name foi alterado.

Exemplo:

```
struct Name
  {
   string            first_name;                 // nome
   string            last_name;                  // último nome
  };

class CPerson
  {
protected:
   Name              m_name;                     // nome
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

Também Veja

Tipos de Dados

© 2000-2024, MetaQuotes Ltd.

Herança

O principal característica da OOP é o incentivo à reutilização de código através de herança. Uma nova classe é feita a partir de uma já existente, que é chamada de classe base. A classe derivada usa os membros da classe base, mas também pode modificar e complementá-los.

Muitos tipos são variações de tipos já existentes. É muitas vezes tedioso desenvolver um novo código para cada um deles. Além disso, um novo código implica em novos erros. A classe derivada herda a descrição da classe base, assim qualquer re-desenvolvimento e re-teste de código é desnecessário. As relações de herança são hierárquicas.

Hierarquia é um método que permite copiar os elementos em toda a sua diversidade e complexidade. Ela introduz a classificação de objetos. Por exemplo, a tabela periódica de elementos tem gases. Eles possuem propriedades inerentes a todos os elementos periódicos.

Gases inertes constituem a próxima importante subclasse. A hierarquia é que o gás inerte, como árgon, é um gás, e um gás, por sua vez, é parte do sistema. Tal hierarquia permite interpretar o comportamento dos gases inertes com facilidade. Sabemos que seus átomos contêm prótons e elétrons, o que é verdade para todos os outros elementos.

Sabemos que eles estão no estado gasoso à temperatura ambiente, como todos os gases. Sabemos que nenhum gás da subclasse de gases inertes entra usualmente em reações químicas com outros elementos, e isso é uma propriedade de todos os gases inertes.

Considere um exemplo de herança de formas geométricas. Para descrever a completa variedade de formas simples (círculos, triângulos, retângulos, quadrados, etc.), a melhor forma é criar uma classe base (ADT), que é o predecessor de todas as classes derivadas.

Vamos criar uma classe base CShape, que contém apenas a maioria dos membros comuns que descrevem a forma. Estes membros descrevem propriedades que são características de qualquer forma - o tipo da forma e as coordenadas do ponto de ancoragem principal.

Exemplo:

```
//--- A classe base da forma (Shape)
class CShape
  {
protected:
   int       m_type;                    // Tipo de Forma
   int       m_xpos;                    // X - coordenada do ponto base
   int       m_ypos;                    // Y - coordenada do ponto base
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // construtor
   void      SetXPos(int x){m_xpos=x;} // define X
   void      SetYPos(int y){m_ypos=y;} // define Y
  };
```

Sem seguida, criamos novas classes derivadas da classe base, nas quais adicionaremos campos necessários, cada um especificando uma certa classe. Para a forma Círculo, é necessário adicionar um membro que contém o valor do raio. A forma Quadrado é caracterizada pelo valor do lado. Portanto, classes derivadas, herdadas da classe base CShape, serão declaradas como se segue:

© 2000-2024, MetaQuotes Ltd.

```
//--- A classe derivada círculo (Circle)
class CCircle : public CShape        // Depois de um dois pontos, nós definimos a clas
  {                                    // a partir da qual a herança é feita
private:
   int             m_radius;           // raio do círculo

public:
                   CCircle(){m_type=1;}// construtor, tipo 1
  };
```

Para a classe do Quadrado, a declaração é semelhante:

```
//--- a classe derivada quadrado (Square)
class CSquare : public CShape        // Depois de um dois pontos, nós definimos a clas
  {                                    // a partir da qual a herança é feita
private:
   int            m_square_side;       // lado do quadrado

public:
                  CSquare(){m_type=2;} // construtor, tipo 2
  };
```

Deve-se notar que enquanto um objeto é criado, o construtor da classe base é chamado primeiro, e então o construtor da classe derivada é chamado. Quando um objeto é destruído, primeiro o destrutor da classe derivada é chamado, e então o destrutor da classe base é chamado.

Assim, ao declarar a maioria do membros gerais na classe base, podemos acrescentar membros adicionais nas classes derivadas que especificam uma classe particular. A herança permite criar bibliotecas de código poderosas que podem ser reutilizadas muitas vezes.

A sintaxe para criar uma classe derivada a partir de uma classe já existente é a seguinte:

```
class class_name :
          (public | protected | private) opt  base_class_name
  {
   declaração de membros de classe
  };
```

Um dos aspectos da classe derivada é a visibilidade (abertura) de seus sucessores membros (herdeiros). As palavras-chave public, protected e private são usadas para indicar quão disponíveis os membros da classe base estarão disponíveis para a classe derivada. A palavra-chave public após um dois pontos no cabeçalho de uma classe derivada indica que os membros protegidos e públicos da classe base CShape devem ser herdados como membros protegidos e públicos da classe derivada CCircle.

Os membros privados da classe base não são disponíveis para a classe derivada. A herança pública também significa que classes derivadas (CCircle e CSquare) são CShapes. Isto é, o Quadrado (CSquare) é uma Forma (CShape), mas a Forma não necessariamente tem que ser um Quadrado.

© 2000-2024, MetaQuotes Ltd.

A classe derivada é uma modificação da classe base, ele herda os membros protegidos e públicos da classe base. Os construtores e destrutores da classe base não podem ser herdados. Além de membros da classe base, novos membros são adicionados em uma classe derivada.

A classe derivada pode incluir a implementação de funções membro, diferentes da classe base. Isso não tem nada a ver com uma sobrecarga, quando o significado de um mesmo nome de uma função pode ser diferente para diferentes assinaturas.

Em herança protegida, membros públicos e protegidos da classe base tornam-se membros protegidos da classe derivada. Em herança privada, os membros públicos e protegidos da classe base tornam-se membros privados da classe derivada.

Em herança protegida e privada, a relação "o objeto de uma classe derivada é objeto da classe base" não é verdade. Os tipos protegido e privado de herança são raros, e cada um deles precisam ser usados com cautela.

Deve ser entendido que o tipo de herança (public, protected ou private) não afeta a forma de **acessar os membros de classes base na hierarquia de herança a partir de uma classe derivada**. Para qualquer tipo de herança, somente membros da classe base declarados com especificadores de acesso public e protected estarão disponíveis fora das classes derivadas. Vamos verificar isso no seguinte exemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Exemplo de classe com alguns tipos de acesso                     |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:             //--- O membro privado não é disponível a partir de classe deriva
   int               m_member;
protected:           //--- O método protegido é disponível a partir da classe base e s
   int               Member(){return(m_member);}
public:              // O construtor de classe é disponível para todos os membros
                     CBaseClass(){m_member=5;return;};
private:             //--- Um método particular para atribuir um valor para m_member
   void              Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| Classe derivada com erros                                        |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // especificação da herença pública pode ser omitido
  {
public:
   void Func() // Na classe derivada, defina uma função com chamadas aos membros da cl
     {
      //--- Uma tentativa de alterar um membro privado da classe base
      m_member=0;        // Erro, o membro privado da classe base não é disponível
```

```
      Member(0);          // Erro, o método privado da classe base não é disponível em
      //--- Leitura do membro da classe base
      Print(m_member);    // Erro, o membro privado da classe base não é disponível
      Print(Member());    // Sem erro, método protegido é acessível a partir da classe
     }
  };
```

No exemplo acima, CBaseClass tem apenas um método público - o construtor. Construtores são chamados automaticamente na criação de um objeto de classe. Portanto, o membro privado m_member e o método protegido Member() não podem ser chamados do lado de fora. Mas no caso de herança pública, o membro Member() da classe base estará disponível a partir de classes derivadas.

No caso de herança protegida, todos os membros da classe base com acessos público e protegido tornam-se protegidos. Isso significa que membros de dados e métodos públicos da classe base, com herança protegida eles passam a ser disponíveis somente a partir de classes derivadas e de suas derivadas seguintes.

```
//+------------------------------------------------------------------+
//| Exemplo de classe com alguns tipos de acesso                     |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- O membro privado não é disponível a partir de classe deriva
   double             m_Pi;
public:               //--- Obtendo e definindo um valor para m_Pi
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // O construtor de classe é disponível para todos os membros
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| Uma classe derivada, em que m_Pi não pode ser modificada         |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // Herança protegida
  {
private:
   double             m_radius;
public:               //--- Métodos públicos na classe derivada
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| Função de inicialização de script                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Ao criar uma classe derivada, o construtor da classe base será chamada automatic
   CProtectedChildClass pt;
//--- Especifica o raio
   pt.SetRadius(10);
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- Se comentar a string abaixo, obteremos um erro na etapa de compilação, já que Se
// pt.SetPI(3);

//--- Agora declare um variável da classe base e tente definir a constante Pi igual a
   CBaseMathClass bc;
   bc.SetPI(10);
//--- Aqui está o resultado
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

O exemplo mostra que os métodos SetPI() e GetPI() na classe base CBaseMathClasse estão abertos e disponíveis para chamadas a partir de qualquer lugar do programa. Mas ao mesmo tempo, para CProtectedChildClasse, que é derivada dela, estes métodos podem ser chamados somente a partir de métodos da classe CProtectedChildClass ou suas classes derivadas.

No caso de herança privada, todos os membros da classe base com acesso público e protegido tornam-se privados, e chamá-los torná-se impossível em herança posterior.

MQL5 não tem herança múltipla.

Também Veja

Estruturas e Classes

© 2000-2024, MetaQuotes Ltd.

Polimorfismo

Polimorfismo é uma oportunidade para diferentes classes de objetos, relacionadas através de herança, de responder de várias formas quando o mesmo elemento de função for chamado. Isso ajuda a criar um mecanismo universal descrevendo o comportamento não apenas da classe base, mas também das classes descendentes.

Vamos continuar a desenvolver uma classe base CShape, e definir uma função membro GetArea(), destinado a calcular a área de uma forma. Em todas as classes descendentes, produzidas por herança a partir da classe base, nós redefinimos esta função de acordo com as regras de cálculo de área de uma forma (shape) particular.

Para um quadrado (classe CSquare), a área é calculada através de seu lado, para um círculo (classe CCircle), a área é expressa através de seu raio, etc. Nós podemos criar um array para armazenas objetos do tipo CShape, no qual tanto objetos da classe base como todos os objetos de classes descendentes podem ser armazenados. Mais adiante, podemos chamar a mesma função para cada elemento do array.

Exemplo:

```
//--- Classe Base
class CShape
  {
protected:
   int            m_type;                // tipo da forma
   int            m_xpos;                // X - coordenada do ponto base
   int            m_ypos;                // Y - coordenada do ponto de base
public:
   void           CShape(){m_type=0;};   // construtor, tipo=0
   int            GetType(){return(m_type);};// retorna o tipo da forma
virtual
   double         GetArea(){return (0); }// retorna a área da forma
  };
```

Agora, todas as classes derivadas têm uma função membro getArea(), que retorna o valor zero. A implementação desta função em cada descendente não será a mesma.

```
//--- A classe derivada Circle
class CCircle : public CShape            // Depois do dois pontos definimos a classe b
  {                                      // a partir do qual a herança é feita
private:
   double         m_radius;              // raio do círculo

public:
   void           CCircle(){m_type=1;};  // construtor, tipo=1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// área do círculo
  };
```

Para a classe Square, a declaração é a mesma:

```
//--- A classe derivada Square
```

© 2000-2024, MetaQuotes Ltd.

```
class CSquare : public CShape            // Depois do dois pontos definimos a classe b
  {                                      // a partir do qual a herança é feita
private:
   double          m_square_side;        // lado do quadrado

public:
   void            CSquare(){m_type=2;}; // construtor, tipo=2
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}// área quadrada
  };
```

Para calcular a área do quadrado e círculo, precisamos dos correspondentes valores de m_radius e m_square_side, por isso nós adicionamos as funções SetRadius() e SetSide() na declaração da correspondente classe.

Assumimos que objetos de diferentes tipos (CCircle e CSquare) derivados do tipo base CShape são usados em nosso programa. Polimorfismo permite criar um array de objetos da classe base CShape, mas ao declarar este array, estes objetos são desconhecidos e o tipo deles é indefinido.

A decisão sobre que tipo de objeto estará contido em cada elemento do array será tomada diretamente durante a execução do programa. Isso envolve a criação dinâmica de objetos das classes apropriadas, e portanto a necessidade do uso de ponteiros de objeto ao invés de objetos.

O operador new é usado para criação dinâmica de objetos. Cada um destes objetos devem ser individualmente e explicitamente excluídos usando o operador delete. Portanto declararemos um array de ponteiros do tipo CShape, e criaremos um objeto de um tipo apropriado para cada elemento (**new Class_Name**), como mostrado no exemplo de script seguinte:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Declararmos um array de ponteiros de objeto do tipo base
   CShape *shapes[5];   // Um array de ponteiros para objetos CShape

//--- Aqui preenchemos o array com objetos derivados
//--- Declaramos um ponteiro para o objeto de tipo CCircle
   CCircle *circle=new CCircle();
//--- Definimos propriedades do objeto usando o ponteiro do círculo
   circle.SetRadius(2.5);
//--- Colocamos o valor do ponteiro em shapes[0]
   shapes[0]=circle;

//--- Criamos um outro objeto CCircle e escrevemos seu ponteiro em shapes[1]
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- Aqui nós intencionalmente "esquecemos" de definir um valor para shapes[2]
//circle=new CCircle();
```

© 2000-2024, MetaQuotes Ltd.

```
//circle.SetRadius(10);
//shapes[2]=circle;

//--- Definimos NULL para o elemento que não é usado
   shapes[2]=NULL;

//--- Criamos um objeto CSquare e escrevemos seu ponteiro em shapes[3]
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- Criamos um objeto CSquare e escrevemos seu ponteiro em shapes[4]
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- Temos um array de ponteiros, obtemos seu tamanho
   int total=ArraySize(shapes);
//--- Passamos em um loop através de todos os ponteiros no array
   for(int i=0; i<5;i++)
     {
      //--- Se o ponteiro no índice especificado é válido
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- Imprimi o tipo e área da forma
         PrintFormat("O objeto do tipo %d tem a área %G",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- Se o ponteiro tem o tipo POINTER_INVALID
      else
        {
         //--- Notificamos um erro
         PrintFormat("Objeto shapes[%d] não foi inicializado! Seu ponteiro pe %s",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- Devemos excluir todos os objetos criados dinamicamente
   for(int i=0;i<total;i++)
     {
      //--- Nós podemos excluir somente objetos com ponteiros do tipo POINTER_DYNAMIC
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- Notificação de exclusão
         PrintFormat("Excluindo shapes[%d]",i);
         //--- Excluímos um objeto por meio de seu ponteiro
         delete shapes[i];
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   }
```

Favor notar que ao excluir um objeto usando o operador delete, o tipo de seu ponteiro deve ser verificado. Somente objetos com ponteiro do tipo POINTER_DYNAMIC podem ser excluídos usando delete. Para ponteiros de outros tipos, um erro será retornado.

Além da redefinição de funções durante herança, o polimorfismo também inclui a implementação de uma mesma função com diferentes conjuntos de parâmetros dentro de uma classe. Isso significa que a classe pode ter várias funções com o mesmo nome, mas com um tipo e/ou conjunto de parâmetros diferentes. Neste caso, o polimorfismo é implementado através de sobrecarga de função.

Também Veja

Standard Library

© 2000-2024, MetaQuotes Ltd.

Sobrecarga (Overload)

Dentro de uma classe é possível definir dois ou mais métodos que usam o mesmo nome, mas tem número diferente de parâmetros. Quando isso ocorre, estes métodos são chamados de sobrecarregados e tal processo é referido como sobrecarga de método.

Sobrecarga de método é uma forma de realização de polimorfismo. Sobrecarga de métodos é realizado em concordância com as mesmas regras da sobrecarga de função.

Se a função chamada não tem correspondência exata, o compilador busca por uma função adequada em três níveis seqüencialmente:

1. busca dentre os métodos da classe;
2. busca dentre os métodos da classe base, consistentemente a partir do ancestral mais próximo até o mais distante.
3. busca entre outras funções.

Se não houver correspondência exata em todos os níveis, mas várias funções adequadas em diferentes níveis foram encontradas, a função encontrada no menor nível é usada. Dentro de um nível, não pode haver mais de uma função adequada.

Também Veja

Recarregando Funções

© 2000-2024, MetaQuotes Ltd.

Funções Virtuais

A palavra-chave virtual é o especificador de função que fornece um mecanismo para selecionar dinamicamente em tempo de execução uma função-membro apropriada entre as funções de classes base e derivadas. Estrutura não podem ter funções virtuais. Pode ser usado para alterar as declarações de funções-membro somente.

A função virtual, assim como uma função comum, deve ter um corpo executável. Ao ser chamada, sua semântica é a mesma que das outras funções.

Uma função virtual pode ser sobreposta (overridden) em um classe derivada. A escolha de qual definição de função deve ser chamada para uma função virtual é feita dinamicamente (em tempo de execução). Um caso típico é quando uma classe base contém uma função virtual, e as classes derivadas têm sua própria versão desta função.

O ponteiro para a classe base pode indicar tanto um objeto da classe base quanto um objeto de uma classe derivada. A escolha da função-membro a ser chamada será executada em tempo de execução e dependerá do tipo do objeto, não do tipo do ponteiro. Se não houver nenhum membro de um tipo derivado, a função virtual da classe base é usada por default.

Destrutores são sempre virtuais, independentemente se elas estão declaradas com a palavra-chave virtual ou não.

Vamos considerar o uso de funções virtuais no exemplo do MT5_Tetris.mq5. A classe base CTetrisShape com a função virtual de desenhar é definjida na inclusão do arquivo MT5_TetisShape.mqh.

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
```

© 2000-2024, MetaQuotes Ltd.

```
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

Mais adiante, para cada classe derivada, esta função é implementada de acordo com as características da classe descendente. Por exemplo, a primeira forma CTetrisShape1 tem sua implementação própria da função Draw():

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- desenhando formato
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- horizontal
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- vertical
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
     }
  }
```

A forma Quadrado é descrita por meio da classe CTetrisShape6 e tem sua implementação própria do método Draw():

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- desenhando formato
```

© 2000-2024, MetaQuotes Ltd.

```
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
     }
  };
```

Dependendo da classe da qual o objeto criado pertence, é chamada a função virtual desta ou daquela classe derivada.

```
void CTetrisField::NewShape()
  {
//--- criando uma dos 7 possíveis formas aleatoriamente
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- desenhar
   m_shape.Draw();
//---
  }
```

Modificador override

O modificador override indica que a função declarada deve substituir o método da classe pai. O uso deste modificador permite evitar erros durante as substituições, por exemplo, durante uma alteração aleatória na assinatura do método. Ou, por exemplo, na classe base, está definido o método func, ele é usado como argumento da variável do tipo int:

© 2000-2024, MetaQuotes Ltd.

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

A seguir, o método é substituído na classe que está sendo herdada:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

No entanto, no caso de haver um erro, o tipo de argumento é alterado de int para short. De fato, aqui acontece uma sobrecarga do método. Agindo em conformidade com o algoritmo de estabelecimento da função sobrecarregada, em determinadas situações, o compilador pode selecionar o método definido na classe base, em vez de escolher o método de substituição.

Para evitar esses erros, é necessário adicionar o modificador override ao método de substituição.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Se, durante a substituição, for alterada a assinatura do método, o compilador não conseguirá encontrar, na classe pai, o método com essa mesma assinatura e emitir o erro de compilação:

```
'CBar::func' method is declared with 'override' specifier but does not override any ba
```

Modificador final

Ao contrário do anterior, o modificador final proíbe a substituição do método em classes de herança. Se a implementação do método for auto-suficiente e concluída na sua totalidade, declare esse fato usando o modificador final, para ele não ser alterado em conseqüência.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

Como no exemplo acima, ao tentar substituir o método usando o modificador final, o compilador irá emitir um erro:

```
'CFoo::func' method declared as 'final' cannot be overridden by 'CBar::func'
see declaration of 'CFoo::func'
```

Veja Também

Standard Library

© 2000-2024, MetaQuotes Ltd.

Membros Estáticos de uma Classe/Estrutura

Membros Estáticos

Os membros de uma classe podem ser declarados usando o modificador de classe de armazenamento static. Esses membros de dados são compartilhados por todos as instâncias desta classe e são armazenados em um lugar. Membros de dados não estáticos são criados para cada variável de objeto de classe.

A incapacidade de declarar membros estáticos de uma classe teria levado a necessidade de declarar estes dados no nível global do programa. Isso quebraria a relação entre os dados e a classe deles, e não seria consistente com o paradigma básico da OOP - juntar dados e métodos para manipulá-los em uma classe. O membro estático permite que dados de classe não específicos a uma particular instância existam no escopo da classe.

Desde que um membro de classe estática não dependa de uma particular instância, a referência a ele é como se segue:

```
class_name::variable
```

onde *class_name* é o nome da classe, e *variable* é o nome do membro da classe.

Como se vê, para acessar o membro estático de uma classe, o operador de resolução de contexto :: é usado. Ao acessar um membro estático de dentro de métodos da classe, o operador de contexto é opcional.

Membro estático de uma classe deve ser explicitamente inicializado com o valor desejado. Para isso, deve ser declarado e inicializado em escopo global. A sequência de inicialização dos membros estáticos corresponderá a sequência de sua declaração em escopo global.

Por exemplo, temos a classe *CParser* usado para separar o texto, e precisamos contar o número total de palavras e caracteres processados. Nós só precisamos declarar os membros de classe necessários como estáticos e inicializá-los no nível global. Então todas as instâncias da classe usarão um contador comum para palavras e caracteres.

```
//+------------------------------------------------------------------+
//| Classe "Analisador de Texto"                                     |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- Construtor e destrutor
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- Inicialização de membros estáticos da classe Parser no nível global
int CParser::s_words=0;
int CParser::s_symbols=0;
```

© 2000-2024, MetaQuotes Ltd.

Um membro de classe estático pode ser declarado com a palavra-chave *const*. Tais constantes estáticas devem ser inicializadas no nível global com a palavra-chave *const*:

```
//+------------------------------------------------------------------+
//| Classe "Stack" para armazenar dados processados                  |
//+------------------------------------------------------------------+
class CStack
  {
public:
                      CStack(void);
                     ~CStack(void){};
...
private:
   static const int  s_max_length; // Capacidade máxima da pilha
  };

//--- Inicialização da constante estática da classe CStack
const int CStack::s_max_length=1000;
```

Ponteiro this

A palavra-chave this denota um ponteiro implicitamente declarado para si mesmo - para uma instância específica da classe, no contexto do qual o método é executado. Pode ser usado somente em métodos não estáticos da classe. O ponteiro this é um membro não estático implícito de qualquer classe.

Em funções estáticas você pode acessar somente membros/métodos estáticos de uma classe.

Métodos Estáticos

Em MQL5, funções membro do tipo estático podem ser usadas. O modificador *static* deve preceder o tipo do retorno de uma função na declaração dentro de uma classe.

```
class CStack
  {
public:
   //--- Construtor e destrutor
                      CStack(void){};
                     ~CStack(void){};
   //--- Capacidade máxima da pilha
   static int         Capacity();
private:
   int                m_length;     // O número de elementos na pilha
   static const int   s_max_length; // Capacidade máxima da pilha
  };
//+------------------------------------------------------------------+
//| Retorna o número máximo de elementos armazenados na pilha        |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//--- Inicializaзro da constante estбtica da classe CStack
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declare CStack type variable
   CStack stack;
//--- call the object's static method
   Print("CStack.s_max_length=",stack.Capacity());
//--- it can also be called the following way, as the method is static and does not re
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

Um método com o modificador const é chamado de constante e não pode alterar membros implícitos de sua classe. A declaração de funções constantes de uma classe e parâmetros constantes é chamado de controle *const-correção*. Através deste controle pode-se assegurar que o compilador garantirá a consistência de valores de objetos e retornará um erro durante a compilação se existir algo errado.

O modificador const é colocado após a lista de argumentos dentro de uma declaração de classe. A definição do lado de fora de uma classe deve também incluir o modificador *const*:

```
//+------------------------------------------------------------------+
//| Class "Retângulo"                                                |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;      // Largura
   double            m_height;     // Altura
public:
   //--- Construtor e destrutor
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- Calculando a área
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| Retorna a área do objeto "Retângulo"                             |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//| Retorna o produto de duas variáveis                              |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Criar um retângulo reto com os lados iguais a 5 e 6
   CRectangle rect(5,6);
//--- Procurar a área do retângulo usando um método constante
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- Procurar o produto de números usando o método estático da classe CRectangle
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Um argumento adicional a favor do uso do controle de constância é o fato de que neste caso, o compilador gera uma otimização especial, por exemplo, coloca um objeto constante na memória de somente-leitura.

Uma função estática não pode ser determinada com o modificador const, porque este modificar garante a constância dos membros de instância quando chamar esta função. Mas, como mencionado acima, a função estática não pode acessar membros de classe não estática.

Também Veja

Variáveis Estáticas, Variáveis, Referências. Modificador & e Palavra-chave this

© 2000-2024, MetaQuotes Ltd.

Templates de Função

Funções sobrecarregadas são normalmente usadas para realizar operações similares em vários tipos de dados. ArraySize() é um exemplo simples de tal função em MQL5. Ela retorna o tamanho de qualquer tipo de array. De fato, esta função de sistema é sobrecarregada e a implementação completa de tal sobrecarga é escondida dos desenvolvedores de aplicativos MQL5:

```
int  ArraySize(
   void&  array[]      // array verificado
   );
```

Isso significa que o compilador da linguagem MQL5 insere a implementação necessária para cada chamada desta função. Por exemplo, é assim que é implementado para arrays de tipo inteiro:

```
int  ArraySize(
   int&  array[]      // array com elementos de tipo int
   );
```

A função ArraySize() pode ser exibida da seguinte forma para array de tipo MqlRates para trabalhar com cotações em formato de dados históricos:

```
int  ArraySize(
   MqlRates&  array[] // array preenchido com valores de tipo MqlRates
   );
```

Assim, é muito conveniente usar a mesma função para trabalhar com tipos diferentes. No entanto, todo trabalho preliminar deve ser realizado - a função necessária deve ser sobrecarregada para todos os tipos de dados com os quais a função deve trabalhar corretamente.

Existe uma solução conveniente. Se operações similares devem ser executadas para cada tipo de dado, é possível usar templates de função. Neste caso, um programador precisa escrever somente uma descrição de template de função. Ao descrever o template de tal forma, nós devemos especificar somente alguns parâmetros formais ao invés de alguns tipos de dados definidos com os quais a função deve trabalhar. O compilador automaticamente gerará várias funções para o apropriado handling de cada tipo baseado nos tipos de argumentos usados ao chamar a função.

A definição de template de função começa com a palavra-chave template seguida pela lista de parâmetros entre chaves angulares. Cada parâmetro formal é precedido pela palavra-chave typename. Tipos de parâmetros formais são tipos incorporados ou definidos pelo usuário. Eles são usados:

- para especificar os tipos de argumentos da função,
- para especificar o tipo de valor de retorno da função,
- para declarar as variáveis dentro da definição da função

O número de parâmetros de template não pode exceder oito. Cada parâmetro formal na definição de template deve aparecer na lista de parâmetros da função pelo menos uma vez. Cada nome de um parâmetro formal deve ser único.

Abaixo está um exemplo de um template de função para buscar do valor mais alto em um array de qualquer tipo numérico (números inteiros ou reais):

© 2000-2024, MetaQuotes Ltd.

```
template<typename T>
T ArrayMax(T &arr[])
  {
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

Este template define a função que encontra o valor mais alto no array passado e retorna este valor como resultado. Lembre-se que a função ArrayMaximum() construída em MQL5 retorna apenas o índice do valor mais alto que pode ser usado para encontrar o valor em si. Por exemplo:

```
//--- criar array
   double array[];
   int size=50;
   ArrayResize(array,size);
//--- preenche com valores aleatórios
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- encontra a posição do valor mais alto no array
   int max_position=ArrayMaximum(array);
//--- agora, obtém o valor mais alto propriamente no array
   double max=array[max_position];
//--- exibe o valor encontrado
   Print("Valor max = ",max);
```

Assim, nós realizamos dois passos para obter obter o valor mais alto no array. Com o template de função ArrayMax(), nós podemos obter o resultado do tipo necessário apenas passando o array de tipo apropriado para esta função. Isso significa que ao invés das últimas duas linhas:

```
//--- encontra a posição do valor mais alto no array
   int max_position=ArrayMaximum(array);
//--- agora, receber o valor mais alto propriamente no array
   double max=array[max_position];
```

nós podemos usar somente uma linha, cujo resultado retornado tem o mesmo tipo que o array passado para a função:

```
//--- encontra o valor mais alto
   double max=ArrayMax(array);
```

Neste caso, o tipo do resultado retornado pela função ArrayMax() corresponderá automaticamente ao tipo do array.

© 2000-2024, MetaQuotes Ltd.

Use a palavra chave typename para obter o tipo do argumento como uma string a fim de criar métodos de propósito geral para trabalhar com vários tipos de dados. Vamos considerar um exemplo específico da função que retorna tipo de dado como uma string:

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: tipo=",GetTypeName(d_value), ",   valor=", d_value);
   Print("i_value: tipo=",GetTypeName(i_value), ",   valor=", i_value);
   Print("trade: tipo=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| Tipo é retornado como uma linha                                  |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- retorna o tipo como uma linha
   return(typename(T));
//---
  }
```

Templates de função podem também serem usados para métodos de classe, por exemplo:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

© 2000-2024, MetaQuotes Ltd.

Os templates de função não deve ser declarados com as palavras-chave export, virtual e #import.

Sobrecarga de funções de modelo

Em alguns casos, pode ser necessário sobrecarregar uma função de modelo. Por exemplo, temos uma função de modelo que grava, no primeiro parâmetro, o valor do segundo parâmetro com ajuda da conversão explícita de tipos. Na linguagem MQL5, é proibida a conversão do tipo string para o tipo bool, nós podemos fazer isto sozinhos, para tal fim, criamos a sobrecarga da função de modelo. Por exemplo:

```
//+------------------------------------------------------------------+
//| Função de modelo                                                 |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Sobrecarga especial para o caso bool+string                      |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

Como resultado da execução deste código, vemos que, para o par int+string, foi usada a função de modelo Assign(), enquanto, após a segunda chamada, para o par bool+string, já foi usada a versão sobrecarregada.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

Veja também

Sobrecarga

© 2000-2024, MetaQuotes Ltd.

Qual é a melhor coisa sobre os modelos?

Os modelos das funções são usados quando é necessário realizar as mesmas operações com diferentes tipos de dados, por exemplo, localizar o elemento máximo na matriz. A principal vantagem da utilização de modelos é que o programador não necessita escrever uma sobrecarga separada para cada tipo. Ou seja, em vez de várias declarações do conjunto de sobrecargas para cada tipo

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

basta escrever uma função de modelo

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

e, em seguida, usá-la em seu código:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Neste caso, o parâmetro formal *T* especifica o tipo de dados utilizados, durante a compilação, ele é substituído pelo tipo real utilizado, ou seja, o compilador gera automaticamente uma função separada para cada tipo, isto é, double, datetime e assim por diante. Da mesma forma, na linguagem MQL5, você pode criar modelos de classes usando todas as vantagens de tal abordagem.

© 2000-2024, MetaQuotes Ltd.

Modelos de classes

O modelo de classe é declarado usando a palavra-chave template, seguida pelos colchetes angulares <>, nos quais são listados os parâmetros formais com a palavra-chave typename. Este registro indica ao compilador que está perante uma classe genérica na com o parâmetro formal *T* que especifica o tipo real da variável ao implementar a classe. Por exemplo, criamos uma classe vector para armazenar a matriz com elementos do tipo *T*:

```
#define TOSTR(x) #x+" "   // macro para exibir o nome do objeto
//+------------------------------------------------------------------+
//| Classe vector para armazenar elementos do tipo T                 |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- por padrão, o construtor cria uma matriz de 10 elementos
   void TArray(void){ArrayResize(m_array,10);}
   //--- construtor para criar um vetor com o tamanho definido da matriz
   void TArray(int size){ArrayResize(m_array,size);}
   //--- retorna o tipo e número de dados que são armazenados no objeto do tipo TArray
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

Em seguida, no programa, criamos de maneiras diferentes três objetos *TArray* para trabalhar com diferentes tipos

```
void OnStart()
  {
   TArray<double> double_array;   // por padrão, o tamanho do vetor é 10
   TArray<int> int_array(15);     // o tamanho do vetor é 15
   TArray<string> *string_array;  // ponteiro para o vetor TArray<string>
//--- criamos o objeto dinâmico
   string_array=new TArray<string>(20);
//--- no Diário, exibimos o nome do objeto, tipo de dados e tamanho do vetor
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- excluímos o objeto dinâmico antes de encerrar o programa
   delete(string_array);
  }
```

Resultado do script:

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

© 2000-2024, MetaQuotes Ltd.

Como resultado, foram criados 3 vetores com diferentes tipos de dados: double, int e string.

Os modelos de classes são adequados para desenvolver recipientes, isto é, os objetos destinados a encapsular qualquer tipo de objeto. Os objetos dos recipientes são coleções que já contêm objetos de um tipo particular. Normalmente, o recipiente imediatamente é integrado e implementado para trabalhar com dados que são armazenados nele.

Por exemplo, é possível criar um modelo de classe que não permita acessar um elemento fora da matriz e, assim, evitar o erro crítico "out of range".

```
//+------------------------------------------------------------------+
// | Classe para acessar com segurança um elemento da matriz          |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- construtor por padrão
   void              TSafeArray(void){}
   //--- construtor para criar a matriz do tamanho especificado
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- tamanho de matriz
   int               Size(void){return(ArraySize(m_array));}
   //--- alteração do tamanho da matriz
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- libertação da matriz
   void              Erase(void){ZeroMemory(m_array);}
   //--- operador de acesso ao elemento da matriz de acordo com o índice
   T                 operator[](int index);
   //--- operador de atribuição para obter imediatamente todos os elementos a partir d
   void              operator=(const T  &array[]); // matriz do tipo T
  };
//+------------------------------------------------------------------+
//| Operação de obtenção do elemento segundo o índice                |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("%s index %d is not in range (0-%d)!",__FUNCTION__,index,max);
      return(invalid_value);
     }
//---
   return(m_array[index]);
```

```
  }
//+------------------------------------------------------------------+
//| Operação de atribuição para a matriz                             |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- o tipo T deve suportar o operador de cópia
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- copiamos a matriz de cotações
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) retornou o código de erro %d",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- criamos o recipiente e colocamos nele a matriz dos valores MqlRates
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- índice nos limites da matriz
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- índice fora dos limites da matriz
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Note-se que, na descrição dos métodos fora da declaração da classe, também é necessário utilizar a declaração de modelo:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   ...
  }
```

Os modelos de classes e funções permitem especificar vários parâmetros formais, separados por vírgulas, por exemplo, coleção Map para armazenar os pares "chave - valor":

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

Veja também

Modelos de funções, Sobrecarga

© 2000-2024, MetaQuotes Ltd.

Classes abstratas e funções meramente virtuais (puras)

As classes abstratas estão projetadas para criar entidades generalizadas, cuja base, no futuro, prevê a criação de classes derivadas mais específicas. Uma classe abstrata é uma classe que pode ser utilizada apenas como classe base para alguma outra classe, por isso é impossível criar um objeto do tipo classe abstrata

Uma classe, que contenha pelo menos uma função meramente virtual, é abstrata. Portanto, as classes derivadas da classe abstrata devem implementar todas as suas funções meramente virtuais, caso contrário, elas serão também classes abstratas.

A função virtual é declarada como "pura" utilizando a sintaxe do especificador de pureza. Consideremos, como exemplo, uma classe CAnimal, criada apenas para fornecer funções comuns - são os próprios objetos do tipo CAnimal que têm um caráter demasiado geral para um uso prático. Assim, a classe CAnimal é uma boa candidata para classe abstrata:

```
class CAnimal
  {
public:
                        CAnimal();      // construtor
   virtual void         Sound() = 0;    // função meramente virtual (pura)
private:
   double               m_legs_count;   // número de patas do animal
  };
```

Aqui a função Sound() é meramente virtual, porque é declarada com o especificador da função meramente virtual PURE (=0).

São funções meramente virtuais apenas aquelas funções virtuais para as quais é exibido o especificador de pureza PURE, nomeadamente: (=NULL) ou (=0). Exemplo de declaração e utilização de uma classe abstrata:

```
class CAnimal
  {
public:
   virtual void         Sound()=NULL;    // PURE method, deve ser substituído no descende
  };
//--- descendente a partir da classe abstrata
class CCat : public CAnimal
 {
public:
  virtual void          Sound() { Print("Myau"); } // PURE é substituída, classe CCat nã
 };

//--- exemplos de utilização incorreta
new CAnimal;          // erro 'CAnimal' - compilador выдаст ошибку "não é possível inst
CAnimal some_animal; // ошибка 'CAnimal' - компилятор выдаст ошибку "não é possível in

//--- exemplos de utilização correta
new CCat;  // sem erros - a classe CCat não é abstrata
CCat cat;  // sem erros - a classe CCat não é abstrata
```

© 2000-2024, MetaQuotes Ltd.

Restrições sobre o uso de classes abstratas

Quando você chamar, com o construtor, uma função virtual pura (direta ou indiretamente), o resultado será incerto.

```
//+------------------------------------------------------------------+
//| Classe básica abstrata                                           |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- função virtual única
   virtual void      Sound(void)=NULL;
   //--- função
   void              CallSound(void) { Sound(); }
   //--- construtor
   CAnimal()
    {
     //--- chamada explícita para um método virtual
     Sound();
     //--- chamada implícita (através de uma terceira função)
     CallSound();
     //--- no construtor e/ou destruidor sempre são chamadas suas funções,
     //--- apesar da virtualidade e substituíção da função chamada no descendente
     //--- se a função chamada for meramente virutal (pura), então,
     //--- a chamada irá resultar no erro de execução: "pure virtual function call"
    }
  };
```

No entanto, os construtores e destruidores de classes abstratas podem chamar outras funções membro.

© 2000-2024, MetaQuotes Ltd.

Espaços de nomes

O namespace (espaço para nome) é uma área especialmente declarada dentro da qual são definidos vários identificadores: variáveis, funções, classes, etc. A. È definido com ajuda da palavra-chave namespace:

```
namespace nome_espaço {
  // lista de definições de funções, de classes e de variáveis
}
```

O uso do namespace permite dividir o namespace global em subespaços. Todos os identificadores no namespace estão disponíveis um para o outro sem especificação. Para acessar membros do namespace de fora, é usado o operador :: (operação de resolução de contexto).

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- trabalhando com o namespace ProjectData
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

Os namespace são usados para organizar o código como grupos lógicos e para evitar conflitos de nomes que podem ocorrer quando são usadas várias bibliotecas em um programa. Nesses casos, cada biblioteca pode ser declarada em seu namespace, para que possa acessar explicitamente as funções e classes necessárias de cada biblioteca.

Um namespace pode ser declarado em vários blocos em um ou em vários arquivos. O compilador reúne todas as partes durante o pré-processamento e namespace resultante contém todos os membros declarados em todas as partes. Suponhamos que implementemos a classe A no arquivo incluído Sample.mqh:

```
//+------------------------------------------------------------------+
//|                                                      Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
```

© 2000-2024, MetaQuotes Ltd.

```
  };
```

Queremos usar essa classe em nosso projeto, mas já temos a classe A. Para poder usar as duas classes e evitar conflitos de identidade, basta agrupar o arquivo incluído em um namespace:

```
//--- declaramos a primeira classe A
class A
  {
public:
                      A() {Print(__FUNCTION__);}
  };
//--- embrulhamos a classe A do arquivo "Sample.mqh" no namespace "Library" para evita
namespace Library
{
#include "Sample.mqh"
}
//--- adicionamos outra classe ao namespace "Library"
namespace Library
{
class B
  {
public:
                      B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- usamos a classe A do namespace global
   A a1;
//--- usamos as classes A e B do namespace "Library"
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Resultado:
   A::A
   Library::A::A
   Library::B::B
*/
```

Os namespace podem ser aninhados. Um namespace aninhado tem acesso ilimitado aos membros do seu espaço pai, mas os membros do espaço pai não têm acesso ilimitado ao espaço para nome aninhado.

```
namespace General
```

© 2000-2024, MetaQuotes Ltd.

```
{
int Func();

namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
}

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
}
```

Namespace global

Se o identificador não for declarado explicitamente no namespace, ele é implicitamente considerado parte do namespace global. Para especificar explicitamente um identificador global, use operador de escopo de resolução sem nome. Isso permite distinguir esse identificador de qualquer outro elemento com o mesmo nome localizado em um namespace diferente. Por exemplo, ao importar uma função:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  Alguma das nossas funções                                       |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- chamada de função impotada
   Print(lib::Func());
//+--- chamada de nossa função
   Print(::Func());
  }
```

Nesse caso, todas as funções importadas da DLL são incluídas no namespace com o mesmo nome. Isso permite que o compilador determine exclusivamente qual função chamar.

Véase también

Variáveis Globais, Variáveis Locais, Visibilidade Escopo e Tempo de Vida de Variáveis, Criação e Exclusão de Objetos

© 2000-2024, MetaQuotes Ltd.

Constantes, Enumeradores e Estruturas

Para simplificar a escrita do programa e tornar os textos do programa mais convenientes para percepção, a linguagem MQL5 fornece constantes e enumeradores padrões pré-definidos. Além disso, estruturas de serviço são usados para armazenar informação.

Constantes padrão são similares a macros e são do tipo int.

As constantes são agrupadas por seus propósitos:

- Constantes de Gráfico são usadas ao trabalhar com gráficos de preços: abertura, navegação, definição de parâmetros;
- Constantes de objetos são destinadas para processar objetos gráficos que podem ser criados e exibidos em gráficos;
- Constantes de indicadores são usadas para trabalhar com indicadores padrões ou customizadas;
- Constantes do estado do ambiente descrevem propriedades do programa MQL5, mostram informações sobre o terminal do cliente, instrumentos financeiros e conta corrente;
- Constantes de negociação permite especificar uma variedade de informações no decorrer da negociação;
- Constantes nomeadas são constantes da linguagem MQL5;
- Estruturas de dados descrevem formatos de armazenamento de dados utilizados;
- Códigos de erros e avisos descrevem mensagens do compilador e respostas do servidor de negociação a solicitações de negociação;
- Constantes de entrada/saída são projetados para trabalhar com funções de arquivo e exibir de mensagens na tela através da função MessageBox().

© 2000-2024, MetaQuotes Ltd.

Constantes de Gráfico

As constantes descrevendo várias propriedades de gráficos são agrupadas na seguinte forma:

- Tipos de eventos – eventos que ocorrem quando se trabalha com gráficos;
- Janelas de tempo (timeframes) de gráfico – períodos incorporados padrão;
- Propriedades de gráfico – identificadores que são usados como parâmetros de funções de gráfico;
- Constantes de posicionamento - valor de um parâmetro da função ChartNavigate();
- Exibição de gráficos - definição da aparência do gráfico.

© 2000-2024, MetaQuotes Ltd.

Tipos de Eventos de Gráficos

Existem 11 tipos de eventos que podem ser processados usando a função pré-definida OnChartEvent(). Para eventos customizados 65535 identificadores são fornecidos na faixa de CHARTEVENT_CUSTOM a CHARTEVENT_CUSTOM_LAST inclusive. Para gerar um evento customizado, a função EventChartCustom() deve ser usada.

ENUM_CHART_EVENT

ID	Descrição
CHARTEVENT_KEYDOWN	Teclada
CHARTEVENT_MOUSE_MOVE	Movimento de mouse, cliques de mouse (se CHART_EVENT_MOUSE_MOVE=true é definido para o gráfico)
CHARTEVENT_MOUSE_WHEEL	Pressionada o rodada a roda do mouse (se definida a propriedade para o gráfico CHART_EVENT_MOUSE_WHEEL=true)
CHARTEVENT_OBJECT_CREATE	Objeto gráfico criado (se CHART_EVENT_OBJECT_CREATE=true é definido para o gráfico)
CHARTEVENT_OBJECT_CHANGE	Propriedade do objeto gráfico alterada via janela de diálogo de propriedades
CHARTEVENT_OBJECT_DELETE	Objeto gráficoexcluído (se CHART_EVENT_OBJECT_DELETE=true é definido para o gráfico)
CHARTEVENT_CLICK	Dar um clique em um gráfico
CHARTEVENT_OBJECT_CLICK	Dar um clique em um objeto gráfico
CHARTEVENT_OBJECT_DRAG	Arrastar e soltar um objeto gráfico
CHARTEVENT_OBJECT_ENDEDIT	Fim de edição de texto no objeto gráfico Edit
CHARTEVENT_CHART_CHANGE	Alteração do tamanho do gráfico ou modificação das propriedades do gráfico através da janela de Propriedades
CHARTEVENT_CUSTOM	O número inicial de um evento de uma faixa de eventos customizados
CHARTEVENT_CUSTOM_LAST	O número final de um evento de uma faixa de eventos customizados

Para cada tipo de evento, os parâmetros de entrada da função OnChartEvent() têm valores definidos que são requeridos para o processamento deste evento. Os eventos e valores passados através destes parâmetros são listados na tabela abaixo.

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de uma teclada	CHARTEVENT_KEYDOWN	código de uma tecla pressionada	Repita a contagem (o número de vezes que a tecla é repetida como um resultado de que o usuário pressiona a tecla)	O valor da string de uma pequena máscara de descrever o estado de botões do teclado
Eventos de mouse(se CHART_EVENT_MOUSE_MOVE=true é definido para o gráfico)	CHARTEVENT_MOUSE_MOVE	a coordenada X	a coordenada Y	O valor de string de uma máscara de bites descrevendo o estado de botões de mouse
Evento da roda do mouse (se, para o gráfico, definida a propriedade CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Sinalizadores de estados de teclas e botões do mouse, coordenadas X e Y do mouse. A descrição é feita no exemplo abaixo	Valor Delta de rolagem de roda de mouse	—
Evento de criação de objeto gráfico(se CHART_EVENT_OBJECT_CREATE=true é definido para o gráfico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome do objeto gráfico criado
Evento de mudança de uma propriedade de objeto via janela de diálogo de propriedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nome do objeto gráfico modificado
Evento de exclusão de objeto gráfico (se CHART_EVENT_OBJECT_DELETE=t	CHARTEVENT_OBJECT_DELETE	—	—	Nome do objeto gráfico excluído

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
rue está definido para o gráfico)				
Evento de um click de mouse no gráfico	CHARTEVENT_CLICK	a coordenada X	a coordenada Y	—
Evento de um click de mouse num objeto gráfico pertencente ao gráfico	CHARTEVENT_OBJECT_CLICK	a coordenada X	a coordenada Y	Nome do objeto gráfico, na qual o evento ocorreu
Evento de um objeto gráfico arrastado usando o mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome do objeto gráfico movido
Evento da edição de texto finalizada na caixa de entrada do objeto gráfico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome do objeto gráfico LabelEdit, cuja edição de texto foi concluída
Evento de alteração do tamanho do gráfico ou modificação das propriedades do gráfico através da janela de Propriedades	CHARTEVENT_CHART_CHANGE	—	—	—
ID do evento de usuário sob N número	CHARTEVENT_CUSTOM+N	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()

Exemplo:

```
#define KEY_NUMPAD_5        12
#define KEY_LEFT            37
#define KEY_UP              38
#define KEY_RIGHT           39
#define KEY_DOWN            40
#define KEY_NUMLOCK_DOWN    98
```

© 2000-2024, MetaQuotes Ltd.

```
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("O expert com nome ",MQL5InfoString(MQL5_PROGRAM_NAME)," está executando");
//--- Ativar eventos de criação de objetos
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- Ativar eventos de exclusão de objetos
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- A atualização forçada das propriedades do gráfico garante a prontidão para o pro
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Funções ChartEvent                                               |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // Identificador de evento
                  const long& lparam,   // Parâmetro de evento de tipo long
                  const double& dparam, // Parâmetro de evento de tipo double
                  const string& sparam  // Parâmetro de evento de tipo string
                  )
  {
//--- o botão esquerdo do mouse foi pressionado no gráfico
   if(id==CHARTEVENT_CLICK)
     {
      Print("As coordenadas do clique do mouse sobre o gráfico são: x = ",lparam,"  y
     }
//--- o mouse foi clicado sobre o objeto gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("O mouse foi clicado sobre o objeto com o nome '"+sparam+"'");
     }
//--- a tecla foi pressionada
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("O KEY_NUMLOCK_LEFT foi pressionado");   break;
         case KEY_LEFT:          Print("O KEY_LEFT foi pressionado");           break;
         case KEY_NUMLOCK_UP:    Print("O KEY_NUMLOCK_UP foi pressionado");     break;
         case KEY_UP:            Print("O KEY_UP foi pressionado");             break;
         case KEY_NUMLOCK_RIGHT: Print("O KEY_NUMLOCK_RIGHT foi pressionado");  break;
```

© 2000-2024, MetaQuotes Ltd.

```
          case KEY_RIGHT:          Print("O KEY_RIGHT foi pressionado");          break;
          case KEY_NUMLOCK_DOWN:   Print("O KEY_NUMLOCK_DOWN foi pressionado");   break;
          case KEY_DOWN:           Print("O KEY_DOWN foi pressionado");           break;
          case KEY_NUMPAD_5:       Print("O KEY_NUMPAD_5 foi pressionado");       break;
          case KEY_NUMLOCK_5:      Print("O KEY_NUMLOCK_5 foi pressionado");      break;
          default:                 Print("Algumas teclas não listadas foram pressionadas
         }
       ChartRedraw();
      }
//--- o objeto foi excluído
   if(id==CHARTEVENT_OBJECT_DELETE)
      {
       Print("O objeto com o nome ",sparam," foi excluído");
      }
//--- o objeto foi criado
   if(id==CHARTEVENT_OBJECT_CREATE)
      {
       Print("O objeto com o nome ",sparam," foi criado");
      }
//--- o objeto foi movido ou suas coordenadas de ponto de ancoragem foram alteradas
   if(id==CHARTEVENT_OBJECT_DRAG)
      {
       Print("O ponto de ancoragem das coordenadas do objeto com o nome ",sparam," foi
      }
//--- o texto na Edição do objeto foi alterado
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      {
       Print("O texto no campo da Edição do objeto com o nome ",sparam," foi alterado")
      }
  }
```

Para o evento CHARTEVENT_MOUSE_MOVE, o parâmetro string **sparam** Contém informações sobre o estado do teclado e botões do mouse:

Bit	Descrição
1	Estado do botão esquerdo do mouse
2	Estado do botão direito do mouse
3	Estado do botão SHIFT
4	Estado do botão CTRL
5	Estado do botão do meio do mouse
6	Estado do primeiro botão extra do mouse
7	Estado do segundo botão extra do mouse

Exemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- habilitar mensagens do CHART_EVENT_MOUSE_MOVE
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
// --- desativar o menu de contexto do gráfico (à direita)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
// --- desativar a cruz (pelo botão do meio)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- a atualização forçada das propriedades do gráfico garante a prontidão para o pro
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse esquerdo
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse direito
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse do meio
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse primeiro X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse segundo X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
//+------------------------------------------------------------------+
//| Funções ChartEvent                                               |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

Para o evento CHARTEVENT_MOUSE_WHEEL, os parâmetros **lparam** e **dparam** contêm informações sobre o estado da tecla **Ctrl**, **Shift**, dos botões do mouse, coordenadas da cursor e dimensão da rolagem da roda do mouse. Para entender melhor o funcionamento deste Expert Advisor no gráfico, role a roda do mouse pressionando os diversos botões e teclas descritos no código.

Exemplo de processamento de evento CHARTEVENT_MOUSE_WHEEL:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//--- ativação da mensagem de rolagem de roda do mouse
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- a atualização forçada das propriedades do gráfico garante a prontidão para o pro
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- analisemos o estado dos botões e da roda do mouse para este evento
      int flg_keys = (int)(lparam>>32);          // sinalizador de estado de teclas Ct
      int x_cursor = (int)(short)lparam;         // coordenada X, em que acontece o ev
      int y_cursor = (int)(short)(lparam>>16);   // coordenada Y, em que acontece o ev
      int delta    = (int)dparam;                // valor total de rolagem da roda, é
      //--- processamos o sinalizador
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
  }
//+------------------------------------------------------------------+ /*
   Exemplo de saída
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
   CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
   CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

Veja Também

Funções de Manipulação de Evento, Trabalhando com eventos

© 2000-2024, MetaQuotes Ltd.

Janela temporal de Gráfico

Todas as janelas de tempo (timeframes) pré-definidas de gráficos têm identificadores únicos. O identificador PERIOD_CURRENT significa o período corrente de um gráfico, no qual um programa MQL5 está rodando.

ENUM_TIMEFRAMES

ID	Descrição
PERIOD_CURRENT	Janela de tempo corrente
PERIOD_M1	1 minuto
PERIOD_M2	2 minutos
PERIOD_M3	3 minutos
PERIOD_M4	4 minutos
PERIOD_M5	5 minutos
PERIOD_M6	6 minutos
PERIOD_M10	10 minutos
PERIOD_M12	12 minutos
PERIOD_M15	15 minutos
PERIOD_M20	20 minutos
PERIOD_M30	30 minutos
PERIOD_H1	1 hora
PERIOD_H2	2 horas
PERIOD_H3	3 horas
PERIOD_H4	4 horas
PERIOD_H6	6 horas
PERIOD_H8	8 horas
PERIOD_H12	12 horas
PERIOD_D1	1 dia
PERIOD_W1	1 semana
PERIOD_MN1	1 mês

Exemplo:

```
   string chart_name="test_Object_Chart";
   Print("Vamos tentar criar um objeto gráfico com o nome ",chart_name);
//--- Se tal objeto não existir - criá-lo
```

© 2000-2024, MetaQuotes Ltd.

```
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
//--- Define o ativo
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- Define a coordenada X do ponto de ancoragem
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- Define a coordenada Y do ponto de ancoragem
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- Define a largura do gráfico
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- Define a altura
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- Define a janela de tempo
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- Define escala (de 0 a 5)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- Desativa a seleção por mouse
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Identificadores de timeseries

Identificadores de timeseries são usados nas funções iHighest() e iLowest(). Pode ser um dos valores da enumeração

ENUM_SERIESMODE

Identificador	Descrição
MODE_OPEN	Preço de abertura
MODE_LOW	Preço mínimo
MODE_HIGH	Preço máximo
MODE_CLOSE	Preço de fechamento
MODE_VOLUME	Volume de ticks
MODE_REAL_VOLUME	Volume real
MODE_SPREAD	Spread

Veja Também

PeriodSeconds, Period, Data e Hora, Visibilidade de objetos

© 2000-2024, MetaQuotes Ltd.

Propriedades de Gráfico

Identificadores de enumeradores ENUM_CHART_PROPERTY são usados como parâmetros de funções para trabalhar com gráficos. A abreviatura de r/o na coluna "Tipo de Propriedade" significa que esta propriedade é somente de leitura e não pode ser alterada. A abreviatura w/o na coluna "Tipo de Propriedade" significa que esta propriedade é somente leitura e não pode ser recebida. Ao acessar certas propriedades, é necessário especificar um parâmetro adicional, que serve para indicar o número da sub-janela do gráfico. 0 significa a janela principal.

As funções que definem as propriedades do gráfico são realmente utilizadas para enviar comandos de alteração no gráfico. Se essas funções forem executadas com êxito, o comando será incluído na fila comum dos eventos do gráfico. As alterações são implementadas para o gráfico ao manipular a fila dos eventos gráficos.

Assim, não espere uma atualização visual imediata do gráfico após chamar essas funções. Geralmente, o gráfico é atualizado automaticamente pelo terminal após os eventos de alteração - uma nova chegada de cotação, redimensionando a janela do gráfico, etc. Use a função ChartRedraw() Para forçar a atualizar do gráfico.

Para funções ChartSetInteger() e ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

ID	Descrição	Tipo de Propriedade
CHART_SHOW	Plotagem do gráfico de preços. Se estiver definido o valor false, será desativada a plotagem de quaisquer atributos do gráfico de preços e serão excluídos todos os recuos das bordas do gráfico, isto é: escalas de tempo e preço, barra de navegação rápida, eventos de Calendário, rótulos de transações, dicas de indicadores e barras, sub-janela de indicadores, histogramas de volumes, etc. A desativação da plotagem é uma solução ideal para criar uma interface personalizada de programa com utilização de recursos gráficos. Os objetos gráficos são plotados sempre independentemente do valor definido da propriedade CHART_SHOW.	bool
CHART_IS_OBJECT	Identificado objeto de "Gráfico" (OBJ_CHART) - retorna true para um objeto gráfico. Retorna false para um gráfico real	bool r/o
CHART_BRING_TO_TOP	Exibe o gráfico no topo de outros gráficos	bool

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
CHART_CONTEXT_MENU	Habilitar/desabilitar o acesso ao menu de contexto pressionando o botão direito do mouse. O valor CHART_CONTEXT_MENU=false desativa apenas o menu de contexto do gráfico, ao acontecer isto, o menu de contexto, para objetos no gráfico, fica disponível.	bool (valor por padrão true)
CHART_CROSSHAIR_TOOL	Habilitar/desabilitar o acesso ao instrumento "Cruz" pressionando o botão do meio do mouse.	bool (valor por padrão true)
CHART_MOUSE_SCROLL	Rolagem do gráfico horizontalmente usando o botão esquerdo do mouse. Rolagem vertical também está disponível se for definido como true o valor de todas as seguintes propriedades: CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR Se CHART_MOUSE_SCROLL=false, a rolagem do gráfico através da roda do mouse estará indisponível.	bool
CHART_EVENT_MOUSE_WHEEL	Envio de todos os programas mql5 para o gráfico de notificações sobre eventos da roda do mouse (CHARTEVENT_MOUSE_WHEEL)	bool (valor por padrão true)
CHART_EVENT_MOUSE_MOVE	Eventos CHARTEVENT_MOUSE_MOVE) envia notificações de movimento de mouse e clique de mouse a todos os programas MQL5 num gráfico	bool
CHART_EVENT_OBJECT_CREATE	Envia uma notificação de um evento de criação de novo objeto (CHARTEVENT_OBJECT_CREATE) para todos os programas MQL5 em um gráfico	bool
CHART_EVENT_OBJECT_DELETE	Envia uma notificação de um evento de exclusão de objeto (CHARTEVENT_OBJECT_DELETE) para todos os programas MQL5 em um gráfico	bool
CHART_MODE	Tipo de gráfico (candles, barras ou linha)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Gráfico de preços no primeiro plano	bool
CHART_SHIFT	Modo de encaixe do gráfico de preço a partir da borda direita	bool

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
CHART_AUTOSCROLL	Modo de movimentação automática para a borda direita do gráfico	bool
CHART_KEYBOARD_CONTROL	Autorização para controlar o gráfico usando as teclas ("Home", "End", "PageUp", "+", "-", "Seta para cima", etc.). Após definir CHART_KEYBOARD_CONTROL=false, é possível desativar o enrolamento e o ajuste de escala do gráfico, no entanto, ao fazer isto, mantem-se a possibilidade de receber eventos, ao clicar essas teclas, em OnChartEvent().	bool
CHART_QUICK_NAVIGATION	Permite ao gráfico interceptar as teclas Space e Enter para ativar a barra de navegação. A barra de navegação rápida é exibida automaticamente abaixo do gráfico, ao clicar duas vezes ou pressionar as teclas Space/Enter. Desse modo, é possível alterar rapidamente o símbolo, timeframe e data da primeira barra visível.	bool
CHART_SCALE	Escala	int de 0 a 5
CHART_SCALEFIX	Modo de escala fixo	bool
CHART_SCALEFIX_11	Modo de escala 1:1	bool
CHART_SCALE_PT_PER_BAR	Escala para ser especificada em pontos por barra	bool
CHART_SHOW_TICKER	Exibição no canto superior esquerdo do ticker do símbolo. Ao definir CHART_SHOW_TICKER com o valor false simultaneamente também define CHART_SHOW_OHLC como false e desativa a exibição de OHLC	bool
CHART_SHOW_OHLC	Exibição no canto superior esquerdo do ticker do símbolo. Ao definir CHART_SHOW_OHLC com o valor true simultaneamente também define CHART_SHOW_TICKER como true e ativa a exibição de ticker	bool
CHART_SHOW_BID_LINE	Exibe valores de preço de venda como uma linha horizontal em um gráfico	bool
CHART_SHOW_ASK_LINE	Exibe valores de preço de compra como uma linha horizontal em um gráfico	bool

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
CHART_SHOW_LAST_LINE	Exibe valores de último preço como uma linha horizontal em um gráfico	bool
CHART_SHOW_PERIOD_SEP	Exibe separadores verticais entre períodos adjacentes	bool
CHART_SHOW_GRID	Exibe grade no gráfico	bool
CHART_SHOW_VOLUMES	Exibe volume no gráfico	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	Descrições pop-up de objetos gráficos	bool
CHART_SHOW_TRADE_HISTORY	Exibição das negociações a partir do histórico de negociação no gráfico como setas de entradas/saídas Veja descrições da opção "Mostrar histórico de negociação" nas configurações da plataforma.	bool
CHART_VISIBLE_BARS	O número de barras no gráfico que podem ser exibidas	int r/o
CHART_WINDOWS_TOTAL	O número total de janelas de gráficos, incluindo sub-janelas de indicadores	int r/o
CHART_WINDOW_IS_VISIBLE	Visibilidade de sub-janelas	bool r/o modificador - número de sub-janela
CHART_WINDOW_HANDLE	Manuseio da janela do gráfico (HWND)	int r/o
CHART_WINDOW_YDISTANCE	A distância entre a moldura superior da sub-janela do indicador e a moldura superior da janela do gráfico principal, ao longo do eixo Y vertical, in pixels. No caso de um evento de mouse, as coordenadas do cursor são passadas em termos das coordenadas da janela do gráfico principal, enquanto que as coordenadas de objetos gráficos em uma sub-janela do indicador são definidas em relação ao canto superior esquerdo da sub-janela. O valor é requerido para converter as coordenadas absolutas do gráfico principal em coordenadas locais da sub-janela para correto trabalho com os objetos gráficos, cujas coordenadas são definidas em relação ao canto superior esquerdo da moldura da sub-janela.	int r/o modificador - número de sub-janela

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
CHART_FIRST_VISIBLE_BAR	Número das primeiras barras visíveis no gráfico. A indexação de barras é a mesma para séries de tempo.	int r/o
CHART_WIDTH_IN_BARS	Largura de gráfico em barras	int r/o
CHART_WIDTH_IN_PIXELS	Largura de gráfico em pixels	int r/o
CHART_HEIGHT_IN_PIXELS	Altura de gráfico em pixels	int modificador - número de sub-janela
CHART_COLOR_BACKGROUND	Cor de fundo do gráfico	color
CHART_COLOR_FOREGROUND	Cor dos eixos, escalas e linha OHLC	color
CHART_COLOR_GRID	Cor da grade	color
CHART_COLOR_VOLUME	Cor dos volumes e níveis de abertura de posição	color
CHART_COLOR_CHART_UP	Cor para a barra de alta, sombras e bordas de candles de alta	color
CHART_COLOR_CHART_DOWN	Cor para a barra de baixa, sombras e bordas de candles de baixa	color
CHART_COLOR_CHART_LINE	Cor de gráfico de linha e cor de candles "Doji" japoneses	color
CHART_COLOR_CANDLE_BULL	Cor de corpo de um candle de alta	color
CHART_COLOR_CANDLE_BEAR	Cor de corpo de um candle de baixa	color
CHART_COLOR_BID	Cor de nível de preço de venda	color
CHART_COLOR_ASK	Cor de nível de preço de compra	color
CHART_COLOR_LAST	Cor de linha do último preço de operação executada	color
CHART_COLOR_STOP_LEVEL	Cor de níveis de ordem de stop (Stop Loss e Take Profit)	color
CHART_SHOW_TRADE_LEVELS	Exibição de níveis de negociação no gráfico (níveis de abertura de posição, Stop Loss, Take Profit e ordens pendentes	bool
CHART_DRAG_TRADE_LEVELS	Permissão para arrastar os níveis de negociação num gráfico com um mouse. O	bool

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
	modo de arrastar é ativado pelo padrão (valor true)	
CHART_SHOW_DATE_SCALE	Exibição da escala de tempo em um gráfico	bool
CHART_SHOW_PRICE_SCALE	Exibição da escala de preço em um gráfico	bool
CHART_SHOW_ONE_CLICK	Showing the "One click trading" panel on a chart	bool
CHART_IS_MAXIMIZED	Janela de gráfico maximizada	bool r/o
CHART_IS_MINIMIZED	Janela de gráfico minimizada	bool r/o
CHART_IS_DOCKED	A janela do gráfico é fixa. Se for definido o valor false, o gráfico pode ser arrastado para fora do terminal	bool
CHART_FLOAT_LEFT	Coordenada esquerda do gráfico desacoplado em relação à tela virtual	int
CHART_FLOAT_TOP	Coordenada superior do gráfico desacoplado em relação à tela virtual	int
CHART_FLOAT_RIGHT	Coordenada direita do gráfico desacoplado em relação à tela virtual	int
CHART_FLOAT_BOTTOM	Coordenada inferior do gráfico desacoplado em relação à tela virtual	int

Para funções ChartSetDouble() e ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

ID	Descrição	Tipo de Propriedade
CHART_SHIFT_SIZE	O tamanho da barra zero encaixada a partir da borda direita em porcentagens	double (de 10 to 50 porcento)
CHART_FIXED_POSITION	Posição fixa do gráfico a partir da borda esquerda em valor percentual. A posição fixa do gráfico é marcada por um pequeno	double

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
	triângulo cinza no eixo do tempo horizontal. Ele é exibido somente se a rolagem automática do gráfico para a direita na chegada de um tick (preço novo) estiver desabilitada (veja a propriedade CHART_AUTOSCROLL). A barra em uma posição fixa permanece no mesmo lugar ao se ampliar e reduzir o gráfico.	
CHART_FIXED_MAX	Máximo fixo do gráfico	double
CHART_FIXED_MIN	Mínimo fixo do gráfico	double
CHART_POINTS_PER_BAR	Escala em pontos por barra	double
CHART_PRICE_MIN	Mínimo do gráfico	double r/o modificador - número de sub-janela
CHART_PRICE_MAX	Máximo do gráfico	double r/o modificador - número de sub-janela

Para funções ChartSetString() e ChartGetString()

ENUM_CHART_PROPERTY_STRING

ID	Descrição	Tipo de Propriedade
CHART_COMMENT	Texto de um comentário em um gráfico	string
CHART_EXPERT_NAME	Nome do conselheiro executado no gráfico com chart_id especificado	string r/o

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo de Propriedade
CHART_SCRIPT_NAME	Nome do script executado no gráfico com chart_id especificado	string r/o

Exemplo:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

Veja Também

Exemplos de como trabalhar com um gráfico

© 2000-2024, MetaQuotes Ltd.

Constantes de Posicionamento

Três identificadores a partir da lista ENUM_CHART_POSITION são os possíveis valores do parâmetro de *posição* para a função ChartNavigate().

ENUM_CHART_POSITION

ID	Descrição
CHART_BEGIN	Começo do gráfico (o preço mais antigo)
CHART_CURRENT_POS	Posição corrente
CHART_END	Final do gráfico (os preços mais recentes)

Exemplo:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navegar falhou. Erro = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2024, MetaQuotes Ltd.

Representação de Gráfico

Os gráficos de preço podem ser exibidos de três maneiras:

- como barras;
- como candles;
- como um linha.

A maneira específica de exibir o gráfico de preço é definida definida pela função ChartSetInteger(chart_handle,CHART_MODE, chart_mode), onde chart_mode é um dos valores da enumeração ENUM_CHART_MODE.

ENUM_CHART_MODE

ID	Descrição
CHART_BARS	Exibe como uma seqüência de barras
CHART_CANDLES	Exibe como candles japoneses
CHART_LINE	Exibe como uma linha desenha por preços de fechamento

Para especificar o modo de exibição de volumes em um gráfico de preços a função ChartSetInteger(chart_handle, CHART_SHOW_VOLUMES, volume_mode) é usada, onde volume_mode é um dos valores da enumeração ENUM_CHART_VOLUME_MODE.

ENUM_CHART_VOLUME_MODE

ID	Descrição
CHART_VOLUME_HIDE	Volumes não são exibidos
CHART_VOLUME_TICK	Volumes de tick (mudança de preço)
CHART_VOLUME_REAL	Volume de negociação

Exemplo:

```
//--- Obtém o manuseio do gráfico corrente
   long handle=ChartID();
   if(handle>0) // Se bem sucedido, customiza adicionalmente
     {
      //--- Desativa auto-rolagem
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- Define o encaixe da borda direita do gráfico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- Exibe como candles
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- Rola por 100 barras a partir do começo do histórico
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- Define o modo de exibição de volume de tick
```

© 2000-2024, MetaQuotes Ltd.

```
    ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
   }
```

Também Veja

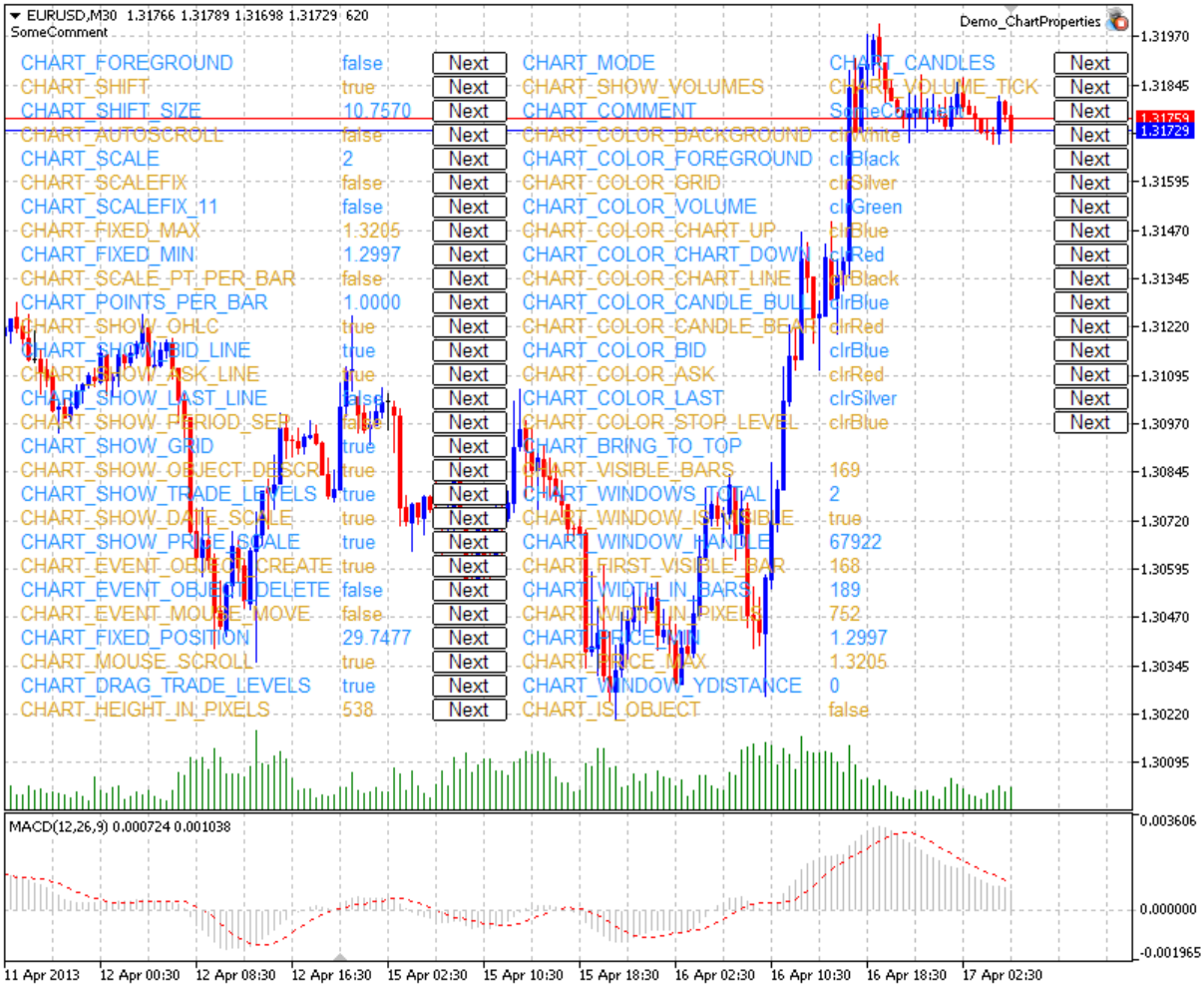
ChartOpen, ChartID

© 2000-2024, MetaQuotes Ltd.

Exemplos de como trabalhar com um gráfico

Exemplos de como trabalhar com propriedades do gráfico. Uma ou duas funções completas são exibidas para cada propriedade. Estas funções permitem configurar/receber o valor da propriedade. Essas funções podem ser utilizadas "como estão" em aplicativos personalizados MQL5.

a imagem abaixo mostra o painel gráfico ilustrando como modificara propriedade do gráficomudando sua aparência. Clicando próximo botão permite definir o novo valor da propriedade apropriada e visualizar as alterações na janela do gráfico.



O código fonte do painel está localizado abaixo.

Propriedades do Gráfico e Exemplos de Funções para Trabalhar com Eles

- **CHART_IS_OBJECT** define se um objeto é um gráfico real ou um objeto gráfico.

```
//+------------------------------------------------------------------------+
//| Definir se um objeto é um gráfico. Se ele for                          |
//| um objeto gráfico, o resultado é verdadeiro. Se ele for um gráfico real |
//| o resultado da variável foi o valor falso.                             |
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- obter a propriedade do gráfico
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      //--- retorna false
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
```

- **CHART_BRING_TO_TOP** exibe o gráfico no topo de outros gráficos.

```
//+------------------------------------------------------------------------+
//| Enviar comando para o terminal para exibir o gráfico acima dos outros.  |
//+------------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- exibe o gráfico no topo de outros gráficos
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_MOUSE_SCROLL** é uma propriedade para rolagem do gráfico usando botão esquerdo do mouse.

```
//+------------------------------------------------------------------------+
//| A função define se a rolagem do gráfico usando o botão esquerdo do mouse |
//| está habilitado.                                                        |
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita a rolagem do gráfico usando botão   |
//| esquerdo do mouse.                                               |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** é uma propriedade de enviar mensagens sobre mudança de eventos e cliques de mouse para aplicações MQL5 CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------+
//| Verificar se mudança de eventos e cliques de mouse               |
//| são enviadas para todas as aplicações mql5 no gráfico.           |
//+------------------------------------------------------------------+
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
```

```
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de envio de mensagens sobre mudança      |
//| de eventos e cliques de mouse para aplicações no                             |
//| gráfico.                                                                     |
//+------------------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE** é uma propriedade de enviar mensagens sobre o evento da criação do objeto gráfico para aplicações MQL5 (CHARTEVENT_OBJECT_CREATE).

```
//+------------------------------------------------------------------------+
//| Verificar se é mensagens do evento da criação de um objeto gráfico     |
//| são enviadas para todas as aplicações mql5 no gráfico.                 |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de envio de mensagens sobre        |
//| o evento da criação do objeto gráfico para aplicações mql5 no          |
//| gráfico.                                                               |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE** é uma propriedade de enviar mensagens sobre o evento da exclusão do objeto gráfico para aplicações MQL5 (CHARTEVENT_OBJECT_DELETE).

```
//+------------------------------------------------------------------------+
//| Verificar se mensagens é do evento da exclusão de um objeto gráfico |
//| são enviadas para todas as aplicações mql5 no gráfico.                |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de envio de mensagens sobre        |
//| o evento da exclusão do objeto gráfico para aplicações mql5 no         |
//| gráfico.                                                               |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_MODE** - tipo de gráfico (candles, barras ou linha).

```
//+------------------------------------------------------------------+
//| Obter tipo de gráfico (candles, barras ou                        |
//| linha).                                                          |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=WRONG_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((ENUM_CHART_MODE)result);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Define tipo de gráfico (candles, barras ou                       |
//| linha).                                                          |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_FOREGROUND** é uma propriedade de exibição do gráfico de preço no primeiro plano.

```
//+------------------------------------------------------------------+
//| A função define se um gráfico de preço é exibido no              |
//| primeiro plano.                                                  |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição do gráfico de preço no |
//| primeiro plano.                                                        |
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHIFT** - modo de encaixe do gráfico de preço a partir da borda direita.

```
//+------------------------------------------------------------------------------
//| A função define se o modo de deslocamento do gráfico de preço do limite direito
//| está habilitado.
//+------------------------------------------------------------------------------
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição do gráfico de preço com  |
//| deslocamento do limite direito.                                          |
//+------------------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_AUTOSCROLL** - modo de movimentação automática para a borda direita do gráfico.

```
//+------------------------------------------------------------------+
//| A função define o modo de auto-rolamento do gráfico              |
//| para a direita no caso de chegada de novos ticks estar habilitado.  |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de auto-rolamento do gráfico |
//| para a direita no caso da chegada de novos ticks.                |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SCALE** - chart scale property.

```
//+------------------------------------------------------------------+
//| Obter escala do gráfico (de 0 a 5).                              |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Definir escala do gráfico (de 0 a 5).                            |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SCALEFIX** - o modo de especificar a escala do gráfico em pontos por barra.

```
//+------------------------------------------------------------------+
//| A função define se o modo de fixar a escala está habilitado.     |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de fixar a escala.           |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SCALEFIX_11** - modo de escala 1:1.

```
//+------------------------------------------------------------------+
//| A função define se a escala "1:1" está habilitada.               |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de escala "1:1"              |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - o modo de especificar a escala do gráfico em pontos por barra.

```
//+------------------------------------------------------------------------------+
//| A função define se o modo de especificar a escala do gráfico é por pontos    |
//| barra está habilitada.                                                       |
//+------------------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------------------
//| A função habilita/desabilita o modo de definir a escala do gráfico em pontos por
//| barra.
//+------------------------------------------------------------------------------------
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_OHLC** - propriedade de exibe valores OHLC no canto superior esquerdo.

```
//+------------------------------------------------------------------+
//| A função define se o modo de exibição dos valores OHLC           |
//| no canto superior esquerdo está habilitado.                      |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição dos valores OHLC no    |
//| canto superior esquerdo do gráfico.                                    |
//+------------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - a propriedade mostra o valor Bid como uma linha horizontal no gráfico.

```
//+-----------------------------------------------------------------------------+
//| A função define se é o modo de exibição da linha do valor Bid no gráfico    |
//| está habilitado.                                                             |
//+-----------------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição da linha Bid no        |
//| gráfico.                                                               |
//+------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - a propriedade de mostrar o valor Ask como uma linha horizontal no gráfico.

```
//+------------------------------------------------------------------+
//| A função define se é o modo de exibição da linha de valor Ask no |
//| gráfico.                                                         |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habibita/desabilita o modo de exibição da linha ASk no  |
//| gráfico.                                                         |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - a propriedade de exibir último valor como uma linha horizontal no gráfico.

```
//+------------------------------------------------------------------------------+
//| A função define se o modo da exibição da linha para a última execução       |
//| do preço de negociação está habilitada.                                      |
//+------------------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------------+
//| A função habilita/desabilita o modo da exibição da linha para a última execução
//| do preço de negociação.
//+------------------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - a propriedade de exibir separadores verticais entre os períodos adjacentes.

```
//+------------------------------------------------------------------+
//| A função define se o modo da exibição dos separadores vertical   |
//| entre os períodos adjacentes está habilitado.                    |
//+------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição dos separadores  |
//| vertical entre os períodos adjacentes.                           |
//+------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_SHOW_GRID** - a propriedade de mostrar a grade do gráfico.

```
//+------------------------------------------------------------------+
//| A função define se a grade do gráfico é exibida.                 |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita a grade do gráfico.                 |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir o valor da propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - a propriedade de mostrar os volumes num gráfico.

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| A função define se os volumes são exibidos no gráfico (ou não são      |
//| mostrados, ticks são mostrados, os atuais são mostrados).              |
//+------------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=WRONG_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+-----------------------------------------------------------------+
//| A função define o mode de exibição dos volumes no gráfico.      |
//+-----------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - a propriedade mostrar descrições pop-up do objeto gráfico.

```
//+-----------------------------------------------------------------+
//| A função define se descrições pop-up de objetos gráficos são    |
//| exibidas quando colocado o cursor do mouse sobre eles.          |
//+-----------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição de descrições pop-up   |
//| de objetos gráficos quando colocado o cursor do mouse sobre eles.      |
//+------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_VISIBLE_BARS** define o número de barras no gráfico que são disponibilizadas para exibição.

```
//+------------------------------------------------------------------------+
//| A função recebe o número de barras que são mostradas (visíveis)        |
//| na janela do gráfico.                                                  |
//+------------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL** define o número total de janelas do gráfico incluindo sub-janelas do indicador.

```
//+------------------------------------------------------------------------+
//| A função obtém o número total de janelas do gráfico incluindo          |
//| subjanelas do indicador.                                               |
//+------------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE** define a visibilidade da sub-janela.

```
//+----------------------------------------------------------------------+
//| A função define se a atual janela do gráfico ou sub-janela           |
//| está visível.                                                        |
//+----------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** retorna o manipulador do gráfico.

```
//+------------------------------------------------------------------+
//| A função obtém o manipulador do gráfico                          |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE** define a distância em pixels entre a quadro superior do sub-janela do indicador e o quadro superior da janela principal do gráfico.

```
//+------------------------------------------------------------------+
//| A função obtém a distância em pixels entre a quadro superior da  |
//| sub-janela e o quadro superior da janela principal do gráfico.   |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR** retorna o número da primeira barra visível no gráfico (barra de indexação que corresponde a série temporal).

```
//+------------------------------------------------------------------------------------
//| A função recebe o número da primeira barra visível no gráfico.
//| Indexação é executada como em séries temporais, as últimas barras têm índices menc
//+------------------------------------------------------------------------------------
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS** retorna a largura do gráfico de barras.

```
//+------------------------------------------------------------------+
//| A função recebe a largura do gráfico de barras.                  |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

© 2000-2024, MetaQuotes Ltd.

- **CHART_WIDTH_IN_PIXELS** retorna a largura da do gráfico em pixels.

```
//+------------------------------------------------------------------+
//| A função recebe a largura da do gráfico em pixels.               |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - propriedade em pixels da altura do gráfico.

```
//+------------------------------------------------------------------+
//| A função recebe o valor da altura do gráfico em pixels.          |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long result=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((int)result);
  }
//+------------------------------------------------------------------+
//| A função define o valor da altura do gráfico em pixels.          |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND -** cor do fundo do gráfico.

```
//+------------------------------------------------------------------+
//| A função recebe a cor do fundo do gráfico.                       |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor do fundo do gráfico
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função define a cor de fundo do gráfico.                       |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir a cor do fundo do gráfico
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - Cor dos eixos, escala e linha OHLC.

```
//+------------------------------------------------------------------+
//| A função recebe a cor dos eixos, da escala e da linha OHLC.      |
//+------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber cor dos eixos, escala e linha OHLC
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor dos eixos, da escala e da linha OHLC.   |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir cor dos eixos, escala e linha OHLC
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_GRID** - cor da grade do gráfico.

```
//+------------------------------------------------------------------+
//| A função recebe a cor da grade do gráfico.                       |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor da grade do gráfico
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor da grade do gráfico.                    |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da grade do gráfico
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - cor dos volumes e níveis de abertura de posição.

```
//+------------------------------------------------------------------+
//| A função recebe cor de volumes e níveis de entrada               |
//| no mercado.                                                      |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- recebe cor de volumes e níveis de entrada no mercado
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor de volumes e entrada                    |
//| no mercado.                                                      |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar cor de volumes e níveis de entrada no mercado
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - cor da barra de alta, sombra e borda da barra de alta (também denominado como castiçal ou vela de alta).

```
//+------------------------------------------------------------------+
//| A função recebe cor da barra de alta, a sua sombra e             |
//| o limite do corpo do candlestick de alta.                        |
//+------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- recebe cor da barra de alta, a sua sombra e o limite do corpo do candlestick de
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura cor da barra de alta, sua sombra e            |
```

© 2000-2024, MetaQuotes Ltd.

```
//| o limite do corpo do candlestick de alta.                          |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da barra de alta, a sua sombra e o limite do corpo do candlest
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - cor da barra de baixa, a sua sombra e limite do corpo do candlestick de baixa.

```
//+------------------------------------------------------------------+
//| A função recebe cor da barra de alta, a sua sombra e              |
//| limite do corpo do candlestick de baixa.                          |
//+------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- recebe cor da barra de baixa, a sua sombra e o limite do corpo do candlestick de
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor da barra de baixa, sua sombra e          |
//| limite do corpo do candlestick de baixa.                          |
//+------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da barra de baixa, sua sombra e e o limite do corpo do candlest
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - Cor da linha do gráfico e barra Doji (também denominado como castiçal ou vela Doji).

```
//+------------------------------------------------------------------------+
//| A função recebe cor da linha do gráfico e candlesticks Doji.           |
//+------------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber cor da linha do gráfico e candlesticks Doji.
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor da linha do gráfico e                   |
//| candlesticks Doji.                                               |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da linha do gráfico e candlesticks Doji
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
```

- **CHART_COLOR_CANDLE_BULL** - cor do corpo da barra de alta (também denominado como castiçal ou vela de alta).

```
//+------------------------------------------------------------------+
//| A função recebe a cor do corpo do candlestick de alta.           |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor do corpo do candlestick de alta
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor do corpo do candlestick de alta.        |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor do corpo do candlestick de alta.
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - cor do corpo da barra de baixa (também denominado como castiçal ou vela de baixa).

```
//+------------------------------------------------------------------+
//| A função recebe a cor do corpo do candlestick de baixa.          |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
color ChartBearColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor do corpo do candlestick de baixa
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor do corpo do candlestick de baixa.       |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor do corpo do candlestick de baixa
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_BID** - Cor de linha de preço de venda.

```
//+------------------------------------------------------------------+
//| A função recebe a cor da linha Bid.                              |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor da linha Bid
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor da linha Bid.                           |
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da linha Bid.
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_ASK** - Cor de linha de preço de compra.

```
//+------------------------------------------------------------------+
//| A função recebe a cor da linha Ask.                              |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor da linha Ask
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor da linha Ask.                           |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da linha Ask
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_LAST** - cor de linha do último preço de operação executada.

```
//+------------------------------------------------------------------+
//| A função recebe a cor da linha do preço do último negócio realizado. |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber cor de linha do último preço de operação executada.
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura a cor do preço do último negócio realizado    |
//| linha.                                                           |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- configurar a cor da linha do preço do último negócio realizado (Last)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sucesso na execução
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - cor de níveis de ordem de stop (Stop Loss e Take Profit).

```
//+------------------------------------------------------------------+
//| A função recebe as cores dos níveis de Stop Loss e Take Profit.  |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- preparar a variável para receber a cor
   long result=clrNONE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber a cor de níveis de ordem de stop (Stop Loss e Take Profit)
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return((color)result);
  }
//+------------------------------------------------------------------+
//| A função configura cores dos níveis de Stop Loss e Take Profit.  |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir a cor de níveis de ordem de stop (Stop Loss e Take Profit)
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - propriedade de exibição de níveis de negociação no gráfico (níveis de abertura de posição, Stop Loss, Take Profit e ordens pendentes.

```
//+------------------------------------------------------------------+
//| A função define se níveis de negociação são exibidos no gráfico. |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibição níveis de negociação |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - propriedade de habilitar a capacidade de arrastar os níveis de negociação no gráfico usando o mouse.

```
//+------------------------------------------------------------------------+
//| Função define se arrastar os níveis de negociação no gráfico com o mouse |
//| é permitido.                                                           |
//+------------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Função habilita/desabilita o modo arrastar os níveis de negócios |
//| no gráfico usando o mouse.                                       |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - propriedade de exibição da escala de tempo em um gráfico.

```
//+------------------------------------------------------------------+
//| A função configura se a escala de tempo é exibida no gráfico.    |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibir a escala de tempo no |
//| gráfico.                                                         |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - propriedade de mostrar a escala de preços no gráfico.

```
//+------------------------------------------------------------------+
//| A função define se a escala de preços é exibida no gráfico.      |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- preparar a variável para obter o valor da propriedade
   long value;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- armazenar o valor da propriedade do gráfico na memória
   result=value;
//--- sucesso na execução
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------------------+
//| A função habilita/desabilita o modo de exibir a escala de preços no          |
//| gráfico.                                                                     |
//+------------------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - Propriedade de exibição do painel "negociar à um clique" num gráfico.

```
//+------------------------------------------------------------------+
//| Verifique se o painel "negociar à um clique" é exibido no gráfico|
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- Preparar a variável para obter o valor da propriedade
   long value;
//--- Redefinir o valor de erro
   ResetLastError();
//--- Receber o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- Exibir a mensagem de erro no diário de Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- Armazenar o valor da propriedade gráfica na memória
   result=value;
//--- Execução bem-sucedida
   return(true);
  }
//+------------------------------------------------------------------+
//| Ativa/desativa a exibição do painel "Negociar à um clique"       |
//| no gráfico                                                       |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- Redefinir o valor de erro
   ResetLastError();
//--- definir o vaolr da propriedade
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- Exibir a mensagem de erro no diário de Experts
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- Execução bem-sucedida
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - o tamanho da barra zero encaixada a partir da borda direita em porcentagens.

```
//+------------------------------------------------------------------------+
//| A função recebe a tamanho do deslocamento da barra zero do limite direito |
//| do gráfico em valores percentuais (de 10% até 50%).                     |
//+------------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
//+------------------------------------------------------------------------------
//| A função configura o tamanho do deslocamento da barra zero da direita
//| limite do gráfico em valores percentuais (de 10% até 50%).
//| Para habilitar o deslocamento modo, valor da propriedade CHART_SHIFT
//| será configurada para true.
//+------------------------------------------------------------------------------
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- exibe uma mensagem para o diário Experts
       Print(__FUNCTION__+", Código de erro = ",GetLastError());
       return(false);
      }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_FIXED_POSITION** - posição fixa do gráfico a partir da borda esquerda em valor percentual.

```
//+------------------------------------------------------------------------+
//| A função recebe a a localização da posição fixa do gráfico de limite   |
//| esquerdo em valor percentual.                                          |
//+------------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
//+---------------------------------------------------------------------+
//| Função configura a localização da posição fixa do gráfico de limite |
//| esquerdo em valor percentual.    Para visualizar a localização da   |
//| posição fixa do gráfico, o valor de propriedade                     |
//| CHART_AUTOSCROLL deve ser definida como falsa.                      |
//+---------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_FIXED_MAX** - propriedade para fixação máxima do gráfico.

```
//+------------------------------------------------------------------+
//| A função recebe o valor de fixação máxima do gráfico.            |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| A função configura o valor de fixação máxima do gráfico.         |
//| Para alterar o valor da propriedade, valor da propriedade        |
//| CHART_SCALEFIX será preliminarmente definida para                |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_FIXED_MIN** - propriedade para fixação mínima do gráfico.

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| A função recebe o valor de fixação mínima do gráfico.            |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| A função configura o valor de fixação mínima do gráfico.         |
//| Para alterar o valor da propriedade, valor da propriedade        |
//| CHART_SCALEFIX será preliminarmente definida para                |
//| true.                                                            |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - valor da escala em pontos por barra.

```
//+------------------------------------------------------------------------+
//| A função recebe a o valor da escala do gráfico em pontos por barra.    |
//+------------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
```

```
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
//+------------------------------------------------------------------+
//| Função configura o valor da escala do gráfico em pontos por barra.   |
//| Para ver o resultado da mudança do valor dessa propriedade,          |
//| o valor de propriedade                                               |
//| CHART_SCALE_PT_PER_BAR deve ser preliminarmente definido como true.  |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_PRICE_MIN** retorna o valor do mínimo do gráfico.

```
//+------------------------------------------------------------------------------+
//| A função recebe a o valor mínimo do gráfico na janela principal, ou numa     |
//| subjanela.                                                                   |
//+------------------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
```

- **CHART_PRICE_MAX** retorna o valor do máximo do gráfico.

```
//+------------------------------------------------------------------------------------+
//| A função recebe a o valor máximo gráfico na janela principal, ou numa              |
//| subjanela.                                                                         |
//+------------------------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- preparar a variável para obter o resultado
   double result=EMPTY_VALUE;
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
     }
//--- retorna o valor da propriedade do gráfico
   return(result);
  }
```

- **CHART_COMMENT** - comentar sobre o gráfico.

```
//+------------------------------------------------------------------------+
//| A função recebe comentário no canto superior esquerdo do gráfico.      |
//+------------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- receber o valor da propriedade
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| A função configura comentário no canto superior esquerda do      |
//| gráfico.                                                         |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- define valor de propriedade
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- exibe uma mensagem para o diário Experts
      Print(__FUNCTION__+", Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - janela do gráfico maximizada

```
//+------------------------------------------------------------------+
//| A função determina se a janela do gráfica está maximizada        |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- preparamos a variável para o valor da propriedade
   long value;
//--- restauramos o valor de erro
   ResetLastError();
//--- obtemos o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- exibimos a mensagem de erro no diário do "Expert"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizamos na variável o valor dos gráficos
   result=value;
//--- implementação bem-sucedida
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - janela de gráfico minimizada

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| função determina se a janela do gráfica está minimizada          |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- preparamos a variável para obter o valor da propriedade
   long value;
//--- restauramos o valor de erro
   ResetLastError();
//--- obtemos o valor da propriedade
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- exibimos a mensagem de erro no diário do "Expert"
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- memorizamos na variável o valor dos gráficos
   result=value;
//--- implementação bem-sucedida
   return(true);
  }
```

Painel para propriedades do gráfico

```
//--- conectar a biblioteca de elementos de controle
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- constantes predefinidas
#define X_PROPERTY_NAME_1    10  // x coordenadas do nome da propriedade na primeira c
#define X_PROPERTY_VALUE_1   225 // x coordenadas do valor da propriedade na primeira
#define X_PROPERTY_NAME_2    345 // x coordenadas do nome da propriedade na segunda e
#define X_PROPERTY_VALUE_2   550 // x coordenadas do valor da propriedade na segunda e
#define X_BUTTON_1           285 // x coordenadas do botão na primeira coluna
#define X_BUTTON_2           700 // x coordenadas do botão na segunda coluna
#define Y_PROPERTY_1         30  // y coordenadas do do início da primeira e segunda c
#define Y_PROPERTY_2         286 // y coordenadas do início da terceira coluna
#define Y_DISTANCE           16  // eixo y da distância entre as linhas
#define LAST_PROPERTY_NUMBER 111 // número da última propriedade gráfica
//--- parâmetros de entrada
input color InpFirstColor=clrDodgerBlue; // Cor de linhas ímpares
input color InpSecondColor=clrGoldenrod; // Cor de linhas pares
//--- variáveis e arrays
CChartObjectLabel  ExtLabelsName[];  // etiquetas para exibir nomes das propriedades
CChartObjectLabel  ExtLabelsValue[]; // etiquetas para exibição dos valores da propri
CChartObjectButton ExtButtons[];     // botões
int                ExtNumbers[];     // índices das propriedades
string             ExtNames[];       // nomes das propriedades
uchar              ExtDataTypes[];   // tipos de dados de propriedade (integer, double
uint               ExtGroupTypes[];  // array que armazena os dados de propriedades pe
uchar              ExtDrawTypes[];   // array que armazena os dados sobre o tipo de ex
double             ExtMaxValue[];    // valores máximos possíveis das propriedades par
double             ExtMinValue[];    // valores mínimos possíveis das propriedades par
```

© 2000-2024, MetaQuotes Ltd.

```
double              ExtStep[];        // passos para alterar as propriedades
int                 ExtCount;         // número total de todas as propriedades
color               ExtColors[2];     // array de cores para exibição das linhas
string              ExtComments[2];   // array de comentários (para propriedade CHART_C
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- exibir a comentário no gráfico
   Comment("SomeComment");
//--- armazenam cores no array para alternarem entre eles mais tarde
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- armazenam comentários no array para alternarem entre eles mais tarde
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- preparar e exibir o painel de controle para o gerenciamento das propriedades do
   if(!PrepareControls())
      return(INIT_FAILED);
//--- sucesso na execução
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de finalização do expert                                  |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- remover o comentário no gráfico
   Comment("");
  }
//+------------------------------------------------------------------+
//| Manipular eventos de um gráfico                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- verificar o evento clicando no objeto do gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- divide o nome do objeto pelo separador
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- verifica se o objeto é um botão
      if(obj_name[0]=="Button")
        {
         //--- receber índice de botão
```

```
         int index=(int)StringToInteger(obj_name[1]);
         //--- desfazer pressionando o botão
         ExtButtons[index].State(false);
         //--- definir o novo valor da propriedade dependendo do seu tipo
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- re-desenhar valores de propriedades
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Alterar a propriedade inteira do gráfico                         |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- receber o valor atual da propriedade
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- definir o seguinte valor da propriedade
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- definir o novo valor de propriedade
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| Alterar a propriedade double do gráfico                          |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- receber o valor atual da propriedade
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- definir o seguinte valor da propriedade
   value=GetNextValue(value,index);
//--- definir o novo valor de propriedade
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Alterar a propriedade string do gráfico                          |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- variável estática para comutação dentro do array ExtComments
   static uint comment_index=1;
//--- alterar o índice para receber outro comentário
   comment_index=1-comment_index;
//--- definir o novo valor de propriedade
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| Definir o próximos valor da propriedade                          |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| Receber a próxima cor para a propriedade tipo color              |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- retornar o seguinte valor de cor
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| Re-desenhar valores de propriedades                              |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- texto de valor da propriedade
   string text;
   long   value;
//--- loop do número de propriedades
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
```

```
        {
         case 'I':
            //--- receber o valor atual da propriedade
            if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
            break;
            //--- texto de propriedade inteiro
            switch(ExtDrawTypes[i])
              {
               //--- propriedade color
               case 'C':
                  text=(string)((color)value);
                  break;
                  //--- propriedade boolean
               case 'B':
                  text=(string)((bool)value);
                  break;
                  //--- propriedade do enumerador ENUM_CHART_MODE
               case 'M':
                  text=EnumToString((ENUM_CHART_MODE)value);
                  break;
                  //--- propriedade do enumerador ENUM_CHART_VOLUME_MODE
               case 'V':
                  text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                  break;
                  //--- número tipo int
               default:
                  text=IntegerToString(value);
                  break;
              }
            break;
         case 'D':
            //--- texto de propriedade double
            text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
            break;
         case 'S':
            //--- texto de propriedade string
            text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
            break;
        }
      //--- mostrar valor de propriedade
      ExtLabelsValue[i].Description(text);
     }
  }
//+------------------------------------------------------------------+
//| Criar o painel para o gerenciamento das propriedades do gráfico  |
//+------------------------------------------------------------------+
bool PrepareControls(void)
  {
//--- alocar memória para arrays com uma reserva
```

```
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- variáveis
   int i=0;      // loop variável
   int col_1=0; // número de propriedades na primeira coluna
   int col_2=0; // número de propriedade na segunda coluna
   int col_3=0; // múmero de propriedade na terceira coluna
//--- número atual de propriedades - 0
   ExtCount=0;
//--- observando propriedades no loop
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- armazenar o número atual de propriedades
      ExtNumbers[ExtCount]=i;
      //--- incrementar o valor da variável do loop
      i++;
      //--- verificar se existe uma propriedade com tal número
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- criar elementos de controle para a propriedade
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- criar etiquetas e um botão para a propriedade
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- número de elementos na primeira coluna foi incrementado
               col_1++;
               break;
            case 2:
               //--- criar etiquetas e um botão para a propriedade
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- número de elementos na segunda coluna foi incrementado
               col_2++;
               break;
            case 3:
               //--- criar apenas etiquetas para a propriedade
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- número de elementos na terceira coluna foi incrementado
               col_3++;
               break;
           }
         //--- definir o máximo e o mínimo do valor e da etapa da propriedade
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- incrementar o número de propriedades
         ExtCount++;
        }
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- liberar a memória não usada pelos arrays
   MemoryAllocation(ExtCount);
//--- re-desenhar valores de propriedades
   RedrawProperties();
   ChartRedraw();
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alocar memória para os arrays                                    |
//+------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------+
//| Verificar se o índice de propriedade pertence a um de            |
//| enumerador ENUM_CHART_PROPERTIES                                 |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- verificar se a propriedade é do tipo integer
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                      // propriedade do enumerador ENUM_CHART_PROP
      GetTypes(ind,group_type,draw_type); // definir parâmetros da propriedade display
      return(true);
     }
//--- verificar se a propriedade é do tipo double
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                      // propriedade do enumerador ENUM_CHART_PROP
      GetTypes(ind,group_type,draw_type); // definir parâmetros da propriedade display
      return(true);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- verificar se a proprieade é do tipo string
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                        // propriedade do enumerador ENUM_CHART_PROI
      GetTypes(ind,group_type,draw_type); // definir parâmetros da propriedade display
      return(true);
     }
//--- propriedade não pertence a qualquer enumeração
   return(false);
  }
//+------------------------------------------------------------------+
//| Definir o grupo que a propriedade será armazenada em             |
//| bem como seu tipo de exibição                                    |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- verificar se a propriedade pertence ao terceiro grupo
//--- propriedades terceiro grupo são apresentados na segunda coluna, a partir de CHAI
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- verificar se a propriedade pertence ao segundo grupo
//--- propriedades segundo grupo são apresentados no início da segunda coluna
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- se você está aqui, a propriedade pertence ao primeiro grupo (primeira coluna)
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+-----------------------------------------------------------------------+
//| A função verifica se a propriedade pertence ao terceiro grupo e       |
//| define o seu tipo display no caso de uma resposta positiva            |
//+-----------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- verificar se a propriedade pertence ao terceiro grupo
   switch(property_number)
     {
      //--- propriedades boolean
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- propriedades inteiro
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
```

© 2000-2024, MetaQuotes Ltd.

```
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- propriedades double
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- na verdade, esta propriedade é um comando para exibir o gráfico em cima
         //--- não há necessidade de aplicar este painel, como a janela sempre será
         //--- em cima das outras antes de a usarmos
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- propriedade não pertencem ao terceiro grupo
      default:
         return(false);
     }
//--- propriedade pertencem ao terceiro grupo
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| A função verifica se a propriedade pertence ao segundo grupo e   |
//| define o seu tipo display no caso de uma resposta positiva       |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- verificar se a propriedade pertence ao segundo grupo
   switch(property_number)
     {
      //--- propriedade de tipo ENUM_CHART_MODE
      case CHART_MODE:
         draw_type='M';
         break;
         //--- propriedade de tipo ENUM_CHART_VOLUME_MODE
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- propriedade string
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- propriedade color
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
```

© 2000-2024, MetaQuotes Ltd.

```
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- propriedade não pertence ao segundo grupo
      default:
         return(false);
     }
//--- propriedade pertence ao segundo grupo
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------------+
//| Esta função é chamada apenas se ela já é conhecida                     |
//| a propriedade não é do segundo e terceiro grupos de propriedades       |
//+------------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- a propriedade pertence ao primeiro grupo
   group_type=1;
//--- definir a propriedade tipo display
   switch(property_number)
     {
      //--- propriedades inteiro
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- propriedades double
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- ficaram apenas as propriedades boolean
      default:
         draw_type='B';
         return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Criar uma etiqueta e um botão para a propriedade                 |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- array estático para comutar dentro do array de cor ExtColors
   static uint color_index[3]={1,1,1};
//--- alterar o índice para receber uma outra cor
   color_index[type]=1-color_index[type];
//--- etiquetas display e um botão (se btn = true) para a propriedade
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| criar um rótulo                                                  |
//+------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Criar o botão                                                    |
//+------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Próximo"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Definir o máximo e o mínimo do valor e da etapa propriedade      |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- definir os valores dependendo do tipo de propriedade
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
```

© 2000-2024, MetaQuotes Ltd.

```
            step=30;
            break;
            //--- valores padrão
        default:
            max=1;
            min=0;
            step=1;
        }
    }
```

© 2000-2024, MetaQuotes Ltd.

Constantes de Objetos

Existem 44 objetos gráficos que podem ser criados e exibidos no gráfico de preços. Todas as constantes para se trabalhar com objetos estão divididos em 9 grupos:

- Tipos de Objetos – Identificadores de objetos gráficos;
- Propriedades de objeto – atribuição e obtenção de propriedades de objetos gráficos;
- Métodos de vinculação de objeto – constantes de posicionamento no gráfico;
- Canto de vinculação – uma indicação do ângulo da tabela, que está posicionada sobre o objeto;
- Visibilidade de objetos – definição de janelas de tempo em que um objeto é visível;
- Níveis de Ondas de Elliott – marcação de gradação de onda;
- Objetos Gann – constantes de tendência para ventilador de Gann e grade de Gann;
- Cores Web – constantes de cores Web pré-definidos;
- Wingdings – códigos de caracteres da fonte Wingdings.

© 2000-2024, MetaQuotes Ltd.

Tipos de Objeto

Quando um objeto gráfico é criado usando a função ObjectCreate(), é necessário especificar o tipo de objeto a ser criado, que pode ser um dos valores da enumeração ENUM_OBJECT. Especificações adicionais de propriedades de objetos são possíveis usando funções para trabalhar com objetos gráficos.

ENUM_OBJECT

ID		Descrição
OBJ_VLINE		Linha Vertical
OBJ_HLINE		Linha Horizontal
OBJ_TREND		Linha de Tendência
OBJ_TRENDBYANGLE		Linha de Tendência por Ângulo
OBJ_CYCLES		Linhas de Ciclo
OBJ_ARROWED_LINE		Linha de Seta
OBJ_CHANNEL		Canal Eqüidistante
OBJ_STDDEVCHANNEL		Canal de Desvio Padrão
OBJ_REGRESSION		Canal de Regressão Linear
OBJ_PITCHFORK		Forquilha de Andrews
OBJ_GANNLINE		Linha Gann
OBJ_GANNFAN		Ventilador Gann
OBJ_GANNGRID		Grade Gann
OBJ_FIBO		Retração de Fibonacci
OBJ_FIBOTIMES		Fusos Horários de Fibonacci
OBJ_FIBOFAN		Fibonacci Fan
OBJ_FIBOARC		Arcos de Fibonacci
OBJ_FIBOCHANNEL		Canal de Fibonacci
OBJ_EXPANSION		Expansão de Fibonacci
OBJ_ELLIOTWAVE5		Onda Motriz de Elliott
OBJ_ELLIOTWAVE3		Onde de Correção de Elliott
OBJ_RECTANGLE		Retângulo
OBJ_TRIANGLE		Triângulo
OBJ_ELLIPSE		Elipse
OBJ_ARROW_THUMB_UP		Polegares para cima

© 2000-2024, MetaQuotes Ltd.

ID		Descrição
OBJ_ARROW_THUMB_DOWN		Polegar para Baixo
OBJ_ARROW_UP		Seta para cima
OBJ_ARROW_DOWN		Seta para baixo
OBJ_ARROW_STOP		Sinal Stop
OBJ_ARROW_CHECK		Conferido
OBJ_ARROW_LEFT_PRICE		Etiqueta de Preço à Esquerda
OBJ_ARROW_RIGHT_PRICE		Rótulo de Preço Direito
OBJ_ARROW_BUY		Comprar
OBJ_ARROW_SELL		Vender
OBJ_ARROW		Seta
OBJ_TEXT		Texto
OBJ_LABEL		Rótulo
OBJ_BUTTON		Botão
OBJ_CHART		Gráfico
OBJ_BITMAP		Bitmap
OBJ_BITMAP_LABEL		Rótulo Bitmap
OBJ_EDIT		Edição
OBJ_EVENT		O objeto "Evento" correspondendo a um evento no calendário econômico
OBJ_RECTANGLE_LABEL		O objeto "Rótulo de Retângulo" para criação e design de interface gráfico customizada.

© 2000-2024, MetaQuotes Ltd.

OBJ_VLINE

Linha vertical.

Observação

Ao desenhar uma linha vertical, é possível definir o modo de exibição de linha para todas as janelas do (propriedade OBJPROP_RAY).

Exemplo

O seguinte script cria e move a linha vertical no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \" Linha Vertical\"."
#property description "Coordenadas do ponto de ancoragem é configurado em porcentagem
#property description "largura da janela do gráfico em barras."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="VLine";      // Nome da Linha
input int             InpDate=25;           // Data do Evento, %
input color           InpColor=clrRed;      // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Estilo da linha
input int             InpWidth=3;           // Largura da linha
input bool            InpBack=false;        // Linha de fundo
input bool            InpSelection=true;    // Destaque para mover
input bool            InpRay=true;          // Continuação da linha para baixo
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpHidden=true;      // Ocultar na lista de objeto
input long            InpZOrder=0;         // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar a linha vertical                                           |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // ID do gráfico
                 const string          name="VLine",      // nome da linha
                 const int             sub_window=0,      // índice da sub-janela
                 datetime              time=0,            // tempo da linha
                 const color           clr=clrRed,        // cor da linha
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo da linha
                 const int             width=1,           // largura da linha
                 const bool            back=false,        // no fundo
                 const bool            selection=true,    // destaque para mover
                 const bool            ray=true,          // continuação da linha para
                 const bool            hidden=true,       //ocultar na lista de objeto
                 const long            z_order=0)         // prioridade para clique do
  {
//--- se o tempo de linha não está definido, desenhá-lo através da última barra
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma linha vertical
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma linha vertical! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir cor da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de exibição da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (verdadeiro) ou desabilitar (falso) o modo de exibição da linha no gra
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover a linha vertical                                           |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // ID do gráfico
               const string name="VLine", // nome da linha
               datetime     time=0)       // tempo da linha
  {
//--- se o tempo de linha não está definido, mover a linha para a última barra
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- mover a linha vertical
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": falhou ao mover a linha vertical! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir a linha vertical                                         |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // ID do gráfico
                 const string name="VLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a linha vertical
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao deletar a linha vertical! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array para armazenar a data de valores a serem utilizados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   datetime date[];
//--- alocação de memória
   ArrayResize(date,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- definir os pontos para desenhar a linha
   int d=InpDate*(bars-1)/100;
//--- criar uma linha vertical
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover a linha
//--- contador de loop
   int h_steps=bars/2;
//--- mover a linha
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.03 segundos de atraso
      Sleep(30);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   VLineDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_HLINE

Linha Horizontal.

Exemplo

O seguinte script cria e move uma linha horizontal no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico \"Linha Horizontal\"."
#property description "Ponto de ancoragem do preço está definido em percentagem da alt
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="HLine";      // Nome da linha
input int             InpPrice=25;          // Preço da linha, %
input color           InpColor=clrRed;      // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Estilo da linha
input int             InpWidth=3;           // Largura da linha
input bool            InpBack=false;        // Linha de fundo
input bool            InpSelection=true;    // Destaque para mover
input bool            InpHidden=true;       // Ocultar na lista de objeto
input long            InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar a linha horizontal                                         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool HLineCreate(const long            chart_ID=0,        // ID de gráfico
                 const string          name="HLine",      // nome da linha
                 const int             sub_window=0,      // índice da sub-janela
                 double                price=0,           // line price
                 const color           clr=clrRed,        // cor da linha
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo da linha
                 const int             width=1,           // largura da linha
                 const bool            back=false,        // no fundo
                 const bool            selection=true,    // destaque para mover
                 const bool            hidden=true,       //ocultar na lista de objeto
                 const long            z_order=0)         // prioridade para clique do
  {
//--- se o preço não está definido, defina-o no atual nível de preço Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um linha horizontal
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar um linha horizontal! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir cor da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de exibição da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover linha horizontal                                           |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // ID do gráfico
```

```
                const string name="HLine", // nome da linha
                double       price=0)      // preço da linha
  {
//--- se o preço não está definido, defina-o no atual nível de preço Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover um linha horizontal
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover um linha horizontal! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir uma linha horizontal                                     |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // ID do gráfico
                 const string name="HLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir uma linha horizontal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao Excluir um linha horizontal! Código de erro = ",GetLastError()
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- tamanho do array de preço
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- array para armazenar data de valores a serem utilizados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   double price[];
//--- alocação de memória
   ArrayResize(price,accuracy);
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a linha
   int p=InpPrice*(accuracy-1)/100;
//--- criar um linha horizontal
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover a linha
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover a linha
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p<accuracy-1)
         p+=1;
      //--- mover o ponto
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a partir do gráfico
   HLineDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
```

© 2000-2024, MetaQuotes Ltd.

```
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TREND

Linha de Tendência.

Observação

Para Linha de Tendência, é possível especificar o modo de continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedadesOBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Exemplo

O script a seguir cria e movimenta a linha de tendência no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico \"Linha de Tendência\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Trend";      // Nome da linha
input int             InpDate1=35;          // Data do 1° ponto, %
input int             InpPrice1=60;         // Preço do 1° ponto, %
input int             InpDate2=65;          // Data do 2° ponto, %
input int             InpPrice2=40;         // Preço do 2° ponto, %
input color           InpColor=clrRed;      // Cor da linha
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo da linha
input int             InpWidth=2;          // Largura da linha
input bool            InpBack=false;       // Linha de fundo
input bool            InpSelection=true;   // Destaque para mover
input bool            InpRayLeft=false;    // Continuação da Linha para a esquerda
input bool            InpRayRight=false;   // Continuação da Linha para a direita
input bool            InpHidden=true;      // Ocultar na lista de objeto
input long            InpZOrder=0;         // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar uma linha de tendência pelas coordenadas fornecidas        |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // ID do gráfico
                 const string          name="TrendLine",  // nome da linha
                 const int             sub_window=0,      // índice da sub-janela
                 datetime              time1=0,           // primeiro ponto de tempo
                 double                price1=0,          // primeiro ponto de preço
                 datetime              time2=0,           // segundo ponto de tempo
                 double                price2=0,          // segundo ponto de preço
                 const color           clr=clrRed,        // cor da linha
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo da linha
                 const int             width=1,           // largura da linha
                 const bool            back=false,        // no fundo
                 const bool            selection=true,    // destaque para mover
                 const bool            ray_left=false,    // continuação da linha para
                 const bool            ray_right=false,   // continuação da linha para
                 const bool            hidden=true,       //ocultar na lista de objeto
                 const long            z_order=0)         // prioridade para clique do
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma linha de tendência pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma linha de tendência! Código de erro = ",GetLastError
      return(false);
     }
//--- definir cor da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de exibição da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
```

© 2000-2024, MetaQuotes Ltd.

```
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover o ponto de ancoragem da linha de tendência                 |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID do gráfico
                      const string name="TrendLine", // nome da linha
                      const int    point_index=0,    // índice do ponto de ancoragem
                      datetime     time=0,           // coordenada do ponto de ancorag
                      double       price=0)          // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem da linha de tendência
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função exclui a linha de tendência do gráfico.                 |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID do gráfico
                 const string name="TrendLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
```

```
//--- excluir uma linha de tendência
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir uma linha de tendência! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verifique os valores de pontos de ancoragem da linha de tendência|
//| e definir padrões valores padrão para aqueles vazios             |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do segundo ponto não está definido, é igual ao primeiro ponto
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem da linha
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a linha
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar uma linha de tendência
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem da linha
//--- contador de loop
   int v_steps=accuracy/5;
//--- mover o primeiro ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- mover o segundo ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2<accuracy-1)
         p2+=1;
      //--- mover o ponto
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- meio segundo de atraso
   Sleep(500);
//--- contador de loop
   int h_steps=bars/2;
//--- mover ambos os pontos de ancoragem na horizontal, ao mesmo tempo
   for(int i=0;i<h_steps;i++)
     {
      //--- usar os seguintes valores
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deslocar os pontos
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.03 segundos de atraso
      Sleep(30);
     }
//--- 1 segundo de atraso
```

© 2000-2024, MetaQuotes Ltd.

```
   Sleep(1000);
//--- excluir uma linha de tendência
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

Linha de Tendência Por Ângulo.

Observação

Para Linha de Tendência Por Angulo, é possível determinar o modo de continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Tanto ângulo e coordenadas do segundo ponto de ancoragem podem ser usados para definir a inclinação da linha.

Exemplo

O script a seguir cria e movimenta a linha de tendência no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Linha de Tendência Por Ângulo\"
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Trend";      // Nome da linha
input int             InpDate1=50;          // Data do 1° ponto, %
input int             InpPrice1=75;         // Preço do 1° ponto, %
input int             InpAngle=0;           // Ângulo de inclinação da linha
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Estilo da linha
input int             InpWidth=2;           // Largura da linha
input bool            InpBack=false;        // Linha de fundo
input bool            InpSelection=true;    // Destaque para mover
input bool            InpRayLeft=false;     // Continuação da Linha para a esquerda
input bool            InpRayRight=true;     // Continuação da Linha para a direita
input bool            InpHidden=true;       // Ocultar na lista de objeto
input long            InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar uma linha de tendência por ângulo                          |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // ID do gráfico
                        const string          name="TrendLine",  // nome da linha
                        const int             sub_window=0,      // índice da sub-jane
                        datetime              time=0,            // tempo do ponto
                        double                price=0,           // preço do ponto
                        const double          angle=45.0,        // ângulo de inclinac
                        const color           clr=clrRed,        // cor da linha
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo da linha
                        const int             width=1,           // largura da linha
                        const bool            back=false,        // no fundo
                        const bool            selection=true,    // destaque para move
                        const bool            ray_left=false,    // continuação da lin
                        const bool            ray_right=true,    // continuação da lin
                        const bool            hidden=true,       // ocultar na lista d
                        const long            z_order=0)         // prioridade para cl
  {
//--- criar o segundo ponto para facilitar, arrastando a linha de tendência com o mous
   datetime time2=0;
   double   price2=0;
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma linha de tendência usando dois pontos
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": falha ao criar uma linha de tendência! Código de erro = ",GetLastError
      return(false);
     }
//--- alterar o ângulo de inclinação da linha de tendência, quando se muda o ângulo, a
//--- ponto da linha são redefinidos automaticamente de acordo com o novo valor do ânc
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- definir cor da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir estilo de linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar ponto de ancoragem das coordenadas da linha de tendência |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // ID do gráfico
                      const string name="TrendLine", // nome da linha
                      datetime     time=0,           // coordenada do ponto de ancorag
                      double       price=0)          // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem da linha de tendência
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mudar o ângulo de inclinação da linha de tendência               |
//+------------------------------------------------------------------+
```

```
bool TrendAngleChange(const long   chart_ID=0,       // ID do grafico
                      const string name="TrendLine", // nome da linha de tendência
                      const double angle=45)         // inclinação do ângulo da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mudar o ângulo de inclinação da linha de tendência
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": falha ao mudar a inclinação da linha! Código de erro = ",GetLastError()
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir a linha de tendência                                     |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // ID do gráfico
                 const string name="TrendLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir uma linha de tendência
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir uma linha de tendência! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verifique os valores de pontos de ancoragem da linha de tendência|
//| e definir padrões valores padrão para aqueles vazios             |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- coordenadas do segundo ponto auxiliar
//--- o segundo ponto será 9 barras a esquerda e tendo o mesmo preço
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime second_point_time[10];
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem da linha
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a linha
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- criar uma linha de tendência
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover e girar a linha
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o ponto de ancoragem e alterar o ângulo de inclinação da linha
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a partir do gráfico
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CYCLES

Linhas Cycle.

Observação

A distância entre as linhas é definida por coordenadas de tempo de dois pontos de ancoragem do objeto.

Exemplo

O seguinte script cria e move a linha cycle no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "script cria a linha cycle no gráfico."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Cycles";    // Nome do objeto
input int             InpDate1=10;         // Data do 1° ponto, %
input int             InpPrice1=45;        // Preço do 1° ponto, %
input int             InpDate2=20;         // Data do 2° ponto, %
input int             InpPrice2=55;        // Preço do 2° ponto, %
input color           InpColor=clrRed;     // Cor das linhas cycle
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Estilo das linhas cycle
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpWidth=1;         // Largura das linhas cycle
input bool            InpBack=false;      // Objeto de Fundo
input bool            InpSelection=true;  // Destaque para mover
input bool            InpHidden=true;     // Ocultar na lista de objeto
input long            InpZOrder=0;        // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criação das linhas cycle                                         |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,        // ID de gráfico
                  const string          name="Cycles",     // nome do objeto
                  const int             sub_window=0,      // índice da sub-janela
                  datetime              time1=0,           // primeiro ponto de tempo
                  double                price1=0,          // primeiro ponto de preço
                  datetime              time2=0,           // segundo ponto de tempo
                  double                price2=0,          // segundo ponto de preço
                  const color           clr=clrRed,        // cor das linhas cycle
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas cycle
                  const int             width=1,           // largura das linhas cycle
                  const bool            back=false,        // no fundo
                  const bool            selection=true,    // destaque para mover
                  const bool            hidden=true,       // ocultar na lista de obje
                  const long            z_order=0)         // prioridade para clicar n
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar linhas cycle pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha ao criar linhas cycle! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir a cor das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a exibição do estilo das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento das linhas com o mous
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // ID do gráfico
                       const string name="Cycles", // nome do objeto
                       const int    point_index=0, // índice do ponto de ancoragem
                       datetime     time=0,        // coordenada do ponto de ancoragem
                       double       price=0)       // coordenada do ponto de ancoragem
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir as linhas cycle                                          |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // ID do gráfico
                  const string name="Cycles") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir as linhas cycle
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir linhas cycle! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Ver os valores de pontos de ancoragem das linhas cycle e definir |
//| valores padrão para valores vazios                               |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do segundo ponto não está definido, é igual ao primeiro ponto
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem das linhas cycle
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar as linhas cycle
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar uma linha de tendência
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int h_steps=bars/5;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      //--- mover o ponto
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   h_steps=bars/4;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d1<bars-1)
         d1+=1;
      //--- mover o ponto
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROWED_LINE

Linha de seta.

Exemplo

O seguinte script cria e move uma linha de seta no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \" linha Arrowed\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="ArrowedLine"; // Nome da linha
input int             InpDate1=35;           // Data do 1° ponto, %
input int             InpPrice1=60;          // Preço do 1° ponto, %
input int             InpDate2=65;           // Data do 2° ponto, %
input int             InpPrice2=40;          // Preço do 2° ponto, %
input color           InpColor=clrRed;       // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Estilo da linha
input int             InpWidth=2;            // Largura da linha
input bool            InpBack=false;         // Linha de fundo
input bool            InpSelection=true;     // Destaque para mover
input bool            InpHidden=true;        // Ocultar na lista de objeto
input long            InpZOrder=0;           // Prioridade para clicar no mouse
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Criar uma linha de setas pelas coordenadas determinadas          |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // ID do gráfico
                       const string          name="ArrowedLine", // nome da linha
                       const int             sub_window=0,       // índice da sub-jane
                       datetime              time1=0,            // primeiro ponto de
                       double                price1=0,           // primeiro ponto de
                       datetime              time2=0,            // segundo ponto de t
                       double                price2=0,           // segundo ponto de p
                       const color           clr=clrRed,         // cor da linha
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // estilo da linha
                       const int             width=1,            // largura da linha
                       const bool            back=false,         // no fundo
                       const bool            selection=true,     // destacar para move
                       const bool            hidden=true,        // ocultar na lista d
                       const long            z_order=0)          // prioridade para cl
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma linha de setas pelas coordenadas determinadas
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": falha ao criar uma linha de seta! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir cor da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de exibição da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Mover o ponto de ancoragem da linha de seta                      |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,         // ID do gráfico
                            const string name="ArrowedLine", // nome da linha
                            const int    point_index=0,      // índice do ponto de anc
                            datetime     time=0,             // coordenada do ponto de
                            double       price=0)            // coordenada do ponto de
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem da linha
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função remove a linha de setas a partir do gráfico             |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,         // ID do gráfico
                       const string name="ArrowedLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- deletar uma linha de seta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma linha de seta! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Conferir valores dos pontos de ancoragem e definir valores padrão|
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do segundo ponto não está definido, é igual ao primeiro ponto
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem da linha
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

```
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a linha
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar uma linha de seta
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem da linha
//--- contador de loop
   int v_steps=accuracy/5;
//--- mover o segundo ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2<accuracy-1)
         p2+=1;
      //--- mover o ponto
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- mover o primeiro ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- mover o ponto
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- meio segundo de atraso
   Sleep(500);
//--- contador de loop
   int h_steps=bars/2;
//--- mover ambos os pontos de ancoragem na horizontal, ao mesmo tempo
   for(int i=0;i<h_steps;i++)
     {
      //--- usar os seguintes valores
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deslocar os pontos
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.03 segundos de atraso
      Sleep(30);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar uma linha de seta
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CHANNEL

Canal Eqüidistante

Observação

Para um canal eqüidistante, é possível especificar o modo de sua continuação para a direita e/ou esquerda (de acordo com as propriedades (OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT). O modo de preenchimento colorido do canal também pode ser definido.

Exemplo

O seguinte script cria e move um canal eqüidistante no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha um objeto gráfico tipo \"canal eqüidistante\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Channel";    // Nome do canal
input int             InpDate1=25;          // Data do 1° ponto, %
input int             InpPrice1=60;         // Preço do 1° ponto, %
input int             InpDate2=65;          // Data do 2° ponto, %
input int             InpPrice2=80;         // Preço do 2° ponto, %
input int             InpDate3=30;          // Data do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpPrice3=40;        // Preço do 3° ponto, %
input color           InpColor=clrRed;     // Cor do canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo das linhas do canal
input int             InpWidth=2;          // Largura da linha do canal
input bool            InpBack=false;       // Canal de Fundo
input bool            InpFill=false;       // Preenchendo o canal com cor
input bool            InpSelection=true;   // Destaque para mover
input bool            InpRayLeft=false;    // Continuação do canal para a esquerda
input bool            InpRayRight=false;   // Continuação do canal para a direita
input bool            InpHidden=true;      // Ocultar na lista de objeto
input long            InpZOrder=0;         // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar um canal eqüidistante pelas coordenadas dadas              |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // ID de gráfico
                   const string          name="Channel",    // nome do canal
                   const int             sub_window=0,      // índice da sub-janela
                   datetime              time1=0,           // primeiro ponto de tempo
                   double                price1=0,          // primeiro ponto de preço
                   datetime              time2=0,           // segundo ponto de tempo
                   double                price2=0,          // segundo ponto de preço
                   datetime              time3=0,           // terceiro ponto de tempo
                   double                price3=0,          // terceiro ponto de preço
                   const color           clr=clrRed,        // cor do sinal
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas do ca
                   const int             width=1,           // largura das linhas do c
                   const bool            fill=false,        // preenchendo o canal co
                   const bool            back=false,        // no fundo
                   const bool            selection=true,    // destaque para mover
                   const bool            ray_left=false,    // continuação do canal pa
                   const bool            ray_right=false,   // continuação do canal pa
                   const bool            hidden=true,       // ocultar na lista de obj
                   const long            z_order=0)         // prioridade para clique
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um canal pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": falha ao criar um canal eqüidistante! Código de erro = ",GetLastError()
      return(false);
     }
//--- definir a cor do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

```
//--- definir a largura da linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,      // ID do gráfico
                        const string name="Channel", // nome do canal
                        const int    point_index=0,  // índice do ponto de ancoragem
                        datetime     time=0,         // coordenada do ponto de ancorag
                        double       price=0)        // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Excluir o canal                                                  |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // ID do gráfico
                   const string name="Channel") // nome do canal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o canal! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| Ver os valores de pontos de ancoragem do canal e definir valores padrão |
//| para aqueles que estão vazios                                          |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- se o tempo do segundo ponto (a direita)não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto (a esquerda) não está definido, está localizado 9 b
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do terceiro ponto não está definido,ele coincide com o tempo do prime
   if(!time3)
      time3=time1;
//--- se o preço do terceiro ponto não está definido, é igual ao preço do segundo pont
   if(!price3)
      price3=price2;
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do canal
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar o canal eqüidistante
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
```

© 2000-2024, MetaQuotes Ltd.

```
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do canal
//--- contador de loop
   int h_steps=bars/6;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      //--- mover o ponto
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d1>1)
         d1-=1;
      //--- mover o ponto
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
```

© 2000-2024, MetaQuotes Ltd.

```
   int v_steps=accuracy/10;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p3>1)
         p3-=1;
      //--- mover o ponto
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

Desvio Padrão do Canal.

Observação

Para Desvio Padrão do Canal, é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT). O modo de preenchimento colorido do canal também pode ser definido.

Propriedade OBJPROP_DEVIATION é usada para alterar o valor do desvio do canal.

Exemplo

O seguinte script cria e move um Desvio Padrão do Canal no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Desvio Padrão do Canal\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="StdDevChannel";    // Nome do canal
input int             InpDate1=10;                // Data do 1° ponto, %
input int             InpDate2=40;                // Data do 2° ponto, %
input double          InpDeviation=1.0;           // Desvio
input color           InpColor=clrRed;            // Cor do canal
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas do canal
input int             InpWidth=2;                // Largura das linhas do canal
input bool            InpFill=false;             // Preenchendo o canal com cor
input bool            InpBack=false;             // Canal de Fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpRayLeft=false;          // Continuação do Canal para a esquer
input bool            InpRayRight=false;         // Continuação do Canal para a direit
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Regressão Linear do Canal pelas coordenadas dadas          |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long            chart_ID=0,        // ID do gráfico
                         const string          name="Channel",    // nome do canal
                         const int             sub_window=0,      // índice da sub-jan
                         datetime              time1=0,           // primeiro ponto de
                         datetime              time2=0,           // segundo ponto de
                         const double          deviation=1.0,     // desvio
                         const color           clr=clrRed,        // cor do canal
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas
                         const int             width=1,           // largura das linha
                         const bool            fill=false,        // preenchendo o car
                         const bool            back=false,        // no fundo
                         const bool            selection=true,    // destaque para mov
                         const bool            ray_left=false,    // continuação do ca
                         const bool            ray_right=false,   // continuação do ca
                         const bool            hidden=true,       // ocultar na lista
                         const long            z_order=0)         // prioridade para c
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeChannelEmptyPoints(time1,time2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um canal pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar o desvio padrão do canal! Código de erro = ",GetLastErro
      return(false);
     }
//--- definir o valor de desvio que afeta a largura do canal
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- definir a cor do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,      // Id do Gráfico
                              const string name="Channel", // nome do canal
                              const int    point_index=0,  // índice do ponto de ancor
                              datetime     time=0)         // coordenada do ponto de a
  {
//--- se o tempo de ponto não está definido, mover o ponto para o barra atual
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o desvio do canal                                        |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // ID do gráfico
                                  const string name="Channel", // nome do canal
                                  const double deviation=1.0)  // desvio
```

```
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mudar o ângulo de inclinação da linha de tendência
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": falha ao mudar o desvio de canal! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o canal                                                  |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // ID do gráfico
                         const string name="Channel") // nome do canal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o canal! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| Ver os valores de pontos de ancoragem do canal e definir valores padrão |
//| para aqueles que estão vazios                                          |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do canal
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- criar desvio padrão do canal
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpSty
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
```

© 2000-2024, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- agora, o canal horizontal para a direita e expandi-lo
//--- contador de loop
   int h_steps=bars/2;
//--- mover o canal
   for(int i=0;i<h_steps;i++)
     {
      //--- usar os seguintes valores
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- mover os pontos de ancoragem
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   double v_steps=InpDeviation*2;
//--- expandir o canal
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

OBJ_REGRESSION

Regressão Linear do Canal.

Observação

Para Regressão Linear do Canal, é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT). O modo de preenchimento colorido do canal também pode ser definido.

Exemplo

O seguinte script cria e move uma Regressão Linear do Canal no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Regressão Linear do Canal\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Regression"; // Nome do canal
input int             InpDate1=10;          // Data do 1° ponto, %
input int             InpDate2=40;          // Data do 2° ponto, %
input color           InpColor=clrRed;      // Cor do canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Estilo das linhas do canal
input int             InpWidth=2;           // Largura das linhas do canal
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpFill=false;        // Preenchendo o canal com cor
input bool            InpBack=false;        // Canal de fundo
input bool            InpSelection=true;    // Destaque para mover
input bool            InpRayLeft=false;     // Continuação do canal para a esquerda
input bool            InpRayRight=false;    // Continuação do canal para a direita
input bool            InpHidden=true;       // Ocultar na lista de objeto
input long            InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Regressão Linear do Canal pelas coordenadas dadas          |
//+------------------------------------------------------------------+
bool RegressionCreate(const long            chart_ID=0,        // ID do gráfico
                      const string          name="Regression", // nome do canal
                      const int             sub_window=0,      // índice da sub-janela
                      datetime              time1=0,           // primeiro ponto de te
                      datetime              time2=0,           // segundo ponto de ter
                      const color           clr=clrRed,        // cor do canal
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas do
                      const int             width=1,           // largura das linhas d
                      const bool            fill=false,        /// preenchendo o canal
                      const bool            back=false,        // no fundo
                      const bool            selection=true,    // destaque para mover
                      const bool            ray_left=false,    // continuação do canal
                      const bool            ray_right=false,   // continuação do canal
                      const bool            hidden=true,       // ocultar na lista de
                      const long            z_order=0)         // prioridade para clic
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeRegressionEmptyPoints(time1,time2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um canal pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar a regressão linear do canal! Código de erro = ",GetLastE
      return(false);
     }
//--- definir a cor do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
```

© 2000-2024, MetaQuotes Ltd.

```
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,      // ID do gráfico
                           const string name="Channel", // nome do canal
                           const int    point_index=0,  // índice do ponto de ancorage
                           datetime     time=0)         // coordenada do ponto de anco
  {
//--- se o tempo de ponto não está definido, mover o ponto para o barra atual
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o canal                                                  |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,      // ID do gráfico
                      const string name="Channel") // nome do canal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o canal
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": falha ao excluir o canal! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| Ver os valores de pontos de ancoragem do canal e definir valores padrão |
//| para aqueles que estão vazios                                          |
//+------------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do canal
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- criar regressão linear do canal
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, o canal horizontal para a direita
//--- contador de loop
   int h_steps=bars/2;
//--- mover o canal
   for(int i=0;i<h_steps;i++)
     {
      //--- usar os seguintes valores
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- mover os pontos de ancoragem
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
```

© 2000-2024, MetaQuotes Ltd.

```
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_PITCHFORK

Tridente Andrews.

Observação

Para Tridente Andrews, é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Você também pode especificar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move o Tridente Andrews no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico \"Tridente Andrews\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Pitchfork";  // Nome do Tridente
input int             InpDate1=14;          // Data do 1° ponto, %
input int             InpPrice1=40;         // Preço do 1° ponto, %
input int             InpDate2=18;          // Data do 2° ponto, %
input int             InpPrice2=50;         // Preço do 2° ponto, %
input int             InpDate3=18;          // Data do 3° ponto, %
input int             InpPrice3=30;         // Preço do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;       // Cor do tridente
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Estilo de linhas do tridente
input int             InpWidth=1;            // Largura das linhas do tridente
input bool            InpBack=false;         // Tridente de fundo
input bool            InpSelection=true;     // Destaque para mover
input bool            InpRayLeft=false;      // Continuação do Tridente para a esquerda
input bool            InpRayRight=false;     // Continuação do Tridente para a direita
input bool            InpHidden=true;        // Ocultar na lista de objeto
input long            InpZOrder=0;           // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Tridente Andrews pelas coordenadas fornecidas              |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // ID de gráfico
                     const string          name="Pitchfork",  // nome do tridente
                     const int             sub_window=0,      // índice da sub-janela
                     datetime              time1=0,           // primeiro ponto de ter
                     double                price1=0,          // primeiro ponto de pre
                     datetime              time2=0,           // segundo ponto de temp
                     double                price2=0,          // segundo ponto de pre
                     datetime              time3=0,           // terceiro ponto de ter
                     double                price3=0,          // terceiro ponto de pre
                     const color           clr=clrRed,        // cor das linhas do tri
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas do
                     const int             width=1,           // largura das linhas do
                     const bool            back=false,        // no fundo
                     const bool            selection=true,    // destaque para mover
                     const bool            ray_left=false,    // continuação do trider
                     const bool            ray_right=false,   // continuação do trider
                     const bool            hidden=true,       // ocultar na lista de o
                     const long            z_order=0)         // prioridade para cliqu
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- criar tridente Andrews pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": falha ao criar \" Tridente Andrews\"! Código de erro = ",GetLastError()
      return(false);
     }
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover o tridente
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do tri
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do tri
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis do Andrews e seus parâmetros          |
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // número de linhas dos níve
                        double          &values[],        // valores de linhas dos nív
                        color           &colors[],        // cor das linhas dos níveis
                        ENUM_LINE_STYLE &styles[],        // estilo das linhas dos nív
                        int             &widths[],        // largura das linhas dos ní
                        const long      chart_ID=0,       // ID do gráfico
                        const string    name="Pitchfork") // nome do tridente
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Movimentar ponto de ancoragem do Tridente Andrews                |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // ID do gráfico
                          const string name="Pitchfork", // nome do canal
                          const int    point_index=0,    // índice do ponto de ancorag
                          datetime     time=0,           // coordenada do ponto de anc
                          double       price=0)          // coordenada do ponto de anc
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Tridente Andrews                                         |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="Pitchfork") // nome do canal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \" Tridente Andrews\"! Código de erro = ",GetLastError
      return(false);
     }
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem do Tridente Andrews e definir |
//| valores para aqueles vazios                                      |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- se o tempo do segundo ponto (acima direita) não está definido, será na barra atu
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto (a esquerda) não está definido, está localizado 9 b
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 200 pontos a mais do que o
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do terceiro ponto não está definido, ele coincide com o tempo do segu
   if(!time3)
      time3=time2;
//--- se o preço do primeiro ponto não está definido, mover 200 pontos a mais do que o
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
```

© 2000-2024, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem do Tridente Andrew
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o Tridente Andrews
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar o tridente
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do tridente
//--- contador de loop
   int v_steps=accuracy/10;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars/8;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d3<bars-1)
         d3+=1;
      //--- mover o ponto
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy/10;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2>1)
         p2-=1;
      //--- mover o ponto
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a tridente do gráfico
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNLINE

Linha Gann.

Observação

Para Linha Gann, é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Ambos ângulo de Gann com uma escala e as coordenadas do segundo ponto de ancoragem podem ser usados para definir a inclinação da linha.

Exemplo

O seguinte script cria e move o Linha Gann no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Linha Gann\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="GannLine";         // Nome da linha
input int             InpDate1=20;                // Data do 1° ponto, %
input int             InpPrice1=75;               // Preço do 1° ponto, %
input int             InpDate2=80;                // Data do 2° ponto, %
input double          InpAngle=0.0;               // Angulo Gann
```

© 2000-2024, MetaQuotes Ltd.

```
input double          InpScale=1.0;              // Escala
input color           InpColor=clrRed;           // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo da linha
input int             InpWidth=2;                // Largura da linha
input bool            InpBack=false;             // Linha de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpRayLeft=false;          // Continuação da Linha para a esquer
input bool            InpRayRight=true;          // Continuação da Linha para a direit
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Linha Gann pelas coordenadas, ângulo e escala              |
//+------------------------------------------------------------------+
bool GannLineCreate(const long            chart_ID=0,        // ID de gráfico
                    const string          name="GannLine",   // nome da linha
                    const int             sub_window=0,      // índice da sub-janela
                    datetime              time1=0,           // primeiro ponto de temp
                    double                price1=0,          // primeiro ponto de pre
                    datetime              time2=0,           // segundo ponto de tempo
                    const double          angle=1.0,         // angulo Gann
                    const double          scale=1.0,         // escala
                    const color           clr=clrRed,        // cor da linha
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo da linha
                    const int             width=1,           // largura da linha
                    const bool            back=false,        // no fundo
                    const bool            selection=true,    // destaque para mover
                    const bool            ray_left=false,    // continuação da linha p
                    const bool            ray_right=true,    // continuação da linha p
                    const bool            hidden=true,       // ocultar na lista de ob
                    const long            z_order=0)         // prioridade para o clic
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar Linha Gann pelas coordenadas fornecidas
//--- corretas coordenadas do segundo ponto de ancoragem são redefinida
//--- automaticamente após o angulo Gann e/ou as mudanças de escala,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Linha Gann\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- alterar angulo Gann
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- alterar a escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- definir cor da linha
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de exibição da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover as linhas
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição da lin
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem do Linha Gann                           |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,       // ID do gráfico
                         const string name="GannLine", // nome da linha
                         const int    point_index=0,   // índice do ponto de ancoragem
                         datetime     time=0,          // coordenada do ponto de ancor
                         double       price=0)         // coordenada do ponto de ancor
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem da linha
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Alterar angulo Gann                                              |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // ID do gráfico
                         const string name="GannLine", // nome da linha
                         const double angle=1.0)       // angulo Gann
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar angulo Gann
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": falha ao alterar o angulo Gann! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar escala Gann Line                                         |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // ID do gráfico
                         const string name="GannLine", // nome da linha
                         const double scale=1.0)       // escala
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a escala (número de pips por barra)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a escala! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função remove a Linha Gann gráfico                             |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // ID do gráfico
                    const string name="GannLine") // nome da linha
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir linha Gann
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": falha ao excluir \"Linha Gann\"! Código de erro = ",GetLastError());
       return(false);
      }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores pontos de ancoragem da linha Gann e definir          |
//| valores padrão para aqueles vazios                               |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem da linha
   datetime date[];
```

© 2000-2024, MetaQuotes Ltd.

```
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a linha Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- criar Linha Gann
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem da linha e mudar o angulo
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o primeiro ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- meio segundo de atraso
   Sleep(500);
//--- definir o valor atual de angulo Gann (alterar
//--- depois de mover o primeiro ponto de ancoragem)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- contador de loop
   v_steps=accuracy/8;
//--- alterar angulo Gann
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a linha do gráfico
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNFAN

Gann Fan.

Observação

Para Gann Fan, é possível determinar o tipo de tendência a partir enumeração ENUM_GANN_DIRECTION. Ao ajustar o valor da escala(OBJPROP_SCALE), é possível alterar o ângulo de inclinação das linhas do fan.

Exemplo

O seguinte script cria e move o Gann Fan no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico tipo \"Gann Fan\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="GannFan";          // Nome do fan
input int             InpDate1=15;                // Data do 1° ponto, %
input int             InpPrice1=25;               // Preço do 1° ponto, %
input int             InpDate2=85;                // Data do 2° ponto, %
input double          InpScale=2.0;               // Escala
input bool            InpDirection=false;         // Direção da tendência
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Cor da linha fan
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas fan
input int             InpWidth=1;                // Largura das linhas fan
input bool            InpBack=false;             // Canal de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Gann Fan                                                   |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // ID de gráfico
                   const string          name="GannFan",    // nome do fan
                   const int             sub_window=0,      // índice da sub-janela
                   datetime              time1=0,           // primeiro ponto de tempo
                   double                price1=0,          // primeiro ponto de preço
                   datetime              time2=0,           // segundo ponto de tempo
                   const double          scale=1.0,         // escala
                   const bool            direction=true,    // direção da tendência
                   const color           clr=clrRed,        // cor da linha fan
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas fan
                   const int             width=1,           // largura das linhas fan
                   const bool            back=false,        // no fundo
                   const bool            selection=true,    // destaque para mover
                   const bool            hidden=true,       // ocultar na lista de obj
                   const long            z_order=0)         // prioridade para clique
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar Gann Fan pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Gann Fan\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- alterar a escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- mudar a direção de tendência Gann Fan (true - descendente, false - ascendente)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- definir cor do fan
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a exibição do estilo das linhas fan
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha do fan
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover o fan
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem do Gann Fan                             |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // ID do gráfico
                        const string name="GannFan", // nome do fan
                        const int    point_index=0,  // índice do ponto de ancoragem
                        datetime     time=0,         // coordenada do ponto de ancorag
                        double       price=0)        // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem do fan
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar a escala Gann Fan                                        |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // ID do gráfico
                        const string name="GannFan", // nome do fan
                        const double scale=1.0)      // escala
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a escala (número de pips por barra)
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a escala! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar a direção de tendência Gann Fan                          |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,     // ID do gráfico
                            const string name="GannFan", // nome do fan
                            const bool   direction=true) // direção da tendência
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a direção de tendência Gann Fan
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a direção de tendência! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função remove o Gann Fan do gráfico                            |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,     // ID do gráfico
                   const string name="GannFan") // nome do fan
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir Gann Fan
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Gann Fan\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem do Gann Fan e definir        |
//| valores padrão para aqueles vazios                               |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem do fan
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar Gann Fan
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- criar Gann Fan
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do fan
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o primeiro ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1<accuracy-1)
         p1+=1;
      //--- mover o ponto
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar a direção tendência do fan para um descendente
   GannFanDirectionChange(0,InpName,true);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o fan do gráfico
   GannFanDelete(0,InpName);
```

© 2000-2024, MetaQuotes Ltd.

```
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_GANNGRID

Grade Gann.

Observação

Para Grade Gann, é possível determinar o tipo de tendência a partir do enumerador ENUM_GANN_DIRECTION. Ao ajustar o valor da escala(OBJPROP_SCALE), é possível alterar o ângulo de inclinação das linhas do grade.

Exemplo

O seguinte script cria e move o Grade Gann no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico tipo \"Grade Gann\"."
#property description "Coordenadas dos pontos de ancoragem do grade são definidos em p
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="GannGrid";         // Nome do grade
input int             InpDate1=15;                // Data do 1° ponto, %
input int             InpPrice1=25;               // Preço do 1° ponto, %
input int             InpDate2=35;                // Data do 2° ponto, %
input double          InpScale=3.0;               // Escala
input bool            InpDirection=false;         // Direção da tendência
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Cor do grade
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas do grade
input int             InpWidth=1;                // Largura das linhas fan
input bool            InpBack=false;             // Grade de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Grade Gann                                                 |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // ID de gráfico
                    const string          name="GannGrid",   // nome do grade
                    const int             sub_window=0,      // índice da sub-janela
                    datetime              time1=0,           // primeiro ponto de tem
                    double                price1=0,          // primeiro ponto de pre
                    datetime              time2=0,           // segundo ponto de tempo
                    const double          scale=1.0,         // escala
                    const bool            direction=true,    // direção da tendência
                    const color           clr=clrRed,        // cor do grade
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas grad
                    const int             width=1,           // largura das linhas gra
                    const bool            back=false,        // no fundo
                    const bool            selection=true,    // destaque para mover
                    const bool            hidden=true,       // ocultar na lista de ob
                    const long            z_order=0)         // prioridade para o clic
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- redefine o valor de erro
   ResetLastError();
//--- Criar Grade Gann pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Grade Gann\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- alterar a escala (número de pips por barra)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- mudar a direção de tendência Gann Fan (true - descendente, false - ascendente)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- definir cor do grade
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a exibição do estilo das linhas grade
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha do grade
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover o grid
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem do Grade Gann                           |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // ID do gráfico
                         const string name="GannGrid", // nome do grade
                         const int    point_index=0,   // índice do ponto de ancoragem
                         datetime     time=0,          // coordenada do ponto de ancor
                         double       price=0)         // coordenada do ponto de ancor
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover ponto de ancoragem do grade
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar a escala Grade Gann                                      |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // ID do gráfico
                         const string name="GannGrid", // nome do grade
                         const double scale=1.0)       // escala
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a escala (número de pips por barra)
```

```
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a escala! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar a direção de tendência Grade Gann                        |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,      // ID do gráfico
                             const string name="GannGrid", // nome do grade
                             const bool   direction=true)  // direção da tendência
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a direção de tendência Grade Gann
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a direção de tendência! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| A função remove o Gann Fan do gráfico                            |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,      // ID do gráfico
                    const string name="GannGrid") // nome do grade
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir Grade Gann
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Grade Gann\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem do Grade Gann e definir      |
//| valores padrão para aqueles vazios                               |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do grade
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar Grade Gann
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- criar Grade Gann
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do grade
//--- contador de loop
   int v_steps=accuracy/4;
//--- mover o primeiro ponto de ancoragem vertical
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars/4;
//--- mover o segundo ponto de ancoragem vertical
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar a direção tendência do grade para um descendente
   GannGridDirectionChange(0,InpName,true);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o grade do gráfico
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBO

Retração Fibonacci.

Observação

Para "Retração Fibonacci", é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com a propriedade OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Você também pode especificar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move a Retração Fibonacci no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
[parcial]#property description "Script desenha objeto gráfico \"Retração Fibonacci\".'
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboLevels";       // Nome do objeto
input int             InpDate1=10;                // Data do 1° ponto, %
input int             InpPrice1=65;               // Preço do 1° ponto, %
input int             InpDate2=90;                // Data do 2° ponto, %
input int             InpPrice2=85;               // Preço do 2° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Cor do objeto
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo da linha
input int             InpWidth=2;                // Largura da linha
input bool            InpBack=false;             // Objeto de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpRayLeft=false;          // Continuação do objeto para a esque
input bool            InpRayRight=false;         // Continuação do objeto para a direi
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Retração Fibonacci pelas coordenadas fornecidas            |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // ID de gráfico
                      const string          name="FiboLevels", // nome do objeto
                      const int             sub_window=0,      // índice da sub-janela
                      datetime              time1=0,           // primeiro ponto de te
                      double                price1=0,          // primeiro ponto de pr
                      datetime              time2=0,           // segundo ponto de ter
                      double                price2=0,          // segundo ponto de pre
                      const color           clr=clrRed,        // cor do objeto
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha do o
                      const int             width=1,           // largura da linha do
                      const bool            back=false,        // no fundo
                      const bool            selection=true,    // destaque para mover
                      const bool            ray_left=false,    // continuação do objet
                      const bool            ray_right=false,   // continuação do objet
                      const bool            hidden=true,       // ocultar na lista de
                      const long            z_order=0)         // prioridade para clic
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- Criar Retração Fibonacci pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha para criar \"Retração Fibonacci\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir estilo de linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
```

© 2000-2024, MetaQuotes Ltd.

```
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do obj
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do obj
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int             levels,            // número de linhas dos níveis
                   double          &values[],         // valores de linhas dos níveis
                   color           &colors[],         // cor das linhas dos níveis
                   ENUM_LINE_STYLE &styles[],         // estilo das linhas dos níveis
                   int             &widths[],         // largura das linhas dos níveis
                   const long      chart_ID=0,        // ID do gráfico
                   const string    name="FiboLevels") // nome do objeto
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem da Retração Fibonacci                   |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // ID do gráfico
                           const string name="FiboLevels", // nome do objeto
                           const int    point_index=0,     // índice do ponto de ancor
                           datetime     time=0,            // coordenada do ponto de a
                           double       price=0)           // coordenada do ponto de a
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Retração Fibonacci                                       |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // ID do gráfico
                      const string name="FiboLevels") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Retração Fibonacci\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//|Vê valores dos pontos de ancoragem da Retração Fibonacci e        |
```

© 2000-2024, MetaQuotes Ltd.

```
//| definir os valores padrões estabelecidos para os pontos vazio    |
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 200 pontos a mais do que o
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem da Retração Fibona
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar Retração Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar um objeto
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int v_steps=accuracy*2/5;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
      if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy*4/5;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2>1)
         p2-=1;
      //--- mover o ponto
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOTIMES

Fibonacci Zonas de Tempo.

Observação

Para "Fibonacci Zonas de Tempo", é possível determinar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move a Fibonacci Zonas de Tempo no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Fibonacci Zonas de Tempo\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboTimes";        // Nome do objeto
input int             InpDate1=10;                // Data do 1° ponto, %
input int             InpPrice1=45;               // Preço do 1° ponto, %
input int             InpDate2=20;                // Data do 2° ponto, %
input int             InpPrice2=55;               // Preço do 2° ponto, %
input color           InpColor=clrRed;            // Cor do objeto
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo da linha
input int             InpWidth=2;                 // Largura da linha
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Objeto de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+-----------------------------------------------------------------+
//| Criar Fibonacci Zonas de Tempo pelas coordenadas fornecidas     |
//+-----------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // ID de gráfico
                     const string          name="FiboTimes",  // nome do objeto
                     const int             sub_window=0,      // índice da sub-janela
                     datetime              time1=0,           // primeiro ponto de ter
                     double                price1=0,          // primeiro ponto de pre
                     datetime              time2=0,           // segundo ponto de temp
                     double                price2=0,          // segundo ponto de pre
                     const color           clr=clrRed,        // cor do objeto
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha do ob
                     const int             width=1,           // largura da linha do
                     const bool            back=false,        // no fundo
                     const bool            selection=true,    // destaque para mover
                     const bool            hidden=true,       // ocultar na lista de
                     const long            z_order=0)         // prioridade para cliq
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar Fibonacci Zonas de Tempo pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Fibonacci Zonas de Tempo\"! Código de erro = ",GetLast
      return(false);
     }
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir estilo de linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,           // número de linhas dos níve
                        double          &values[],        // valores de linhas dos nív
                        color           &colors[],        // cor das linhas dos níveis
                        ENUM_LINE_STYLE &styles[],        // estilo das linhas dos nív
                        int             &widths[],        // largura das linhas dos ní
                        const long      chart_ID=0,       // ID do gráfico
                        const string    name="FiboTimes") // nome do objeto
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover pontos de ancoragem da Fibonacci Zonas de Tempo            |
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // ID gráfico
                          const string name="FiboTimes", // nome do objeto
                          const int    point_index=0,    // índice do ponto de ancora
                          datetime     time=0,           // coordenada do ponto de an
```

```
                            double       price=0)          // coordenada do ponto de anc
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Fibonacci Zonas de Tempo                                 |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // ID gráfico
                     const string name="FiboTimes") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Fibonacci Zonas de Tempo\"! Código de erro = ",GetLa
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar os valores da Fibonacci Zonas de Tempo e               |
//| definir os valores padrões estabelecidos para aqueles vazios     |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
```

```
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado 2 barras deixadas
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 3 barras
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- definir o primeiro ponto 2 barras a esquerda do segundo
      time2=temp[0];
     }
//--- se o preço do segundo ponto não está definido, é igual ao primeiro ponto
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem da Fibonacci Zonas
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
```

© 2000-2024, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- define pontos para desenhar Fibonacci Zonas de Tempo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar um objeto
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int h_steps=bars*2/5;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      //--- mover o ponto
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   h_steps=bars*3/5;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d1<bars-1)
         d1+=1;
      //--- mover o ponto
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOFAN

Fibonacci Fan.

Observação

Para "Fibonacci Fan", é possível determinar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move o Fibonacci Fan no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o objeto gráfico tipo \"Fibonacci Fan\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboFan";          // Nome do fan
input int             InpDate1=10;                // Data do 1° ponto, %
input int             InpPrice1=25;               // Preço do 1° ponto, %
input int             InpDate2=30;                // Data do 2° ponto, %
input int             InpPrice2=50;               // Preço do 2° ponto, %
input color           InpColor=clrRed;            // Cor da linha fan
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo da linha
input int             InpWidth=2;                 // Largura da linha
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpBack=false;             // Objeto de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Fibonacci Fan pelas coordenadas fornecidas                 |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // ID de gráfico
                   const string          name="FiboFan",    // nome do fan
                   const int             sub_window=0,      // índice da sub-janela
                   datetime              time1=0,           // primeiro ponto de tempo
                   double                price1=0,          // primeiro ponto de preço
                   datetime              time2=0,           // segundo ponto de tempo
                   double                price2=0,          // segundo ponto de preço
                   const color           clr=clrRed,        // cor da linha fan
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha fan
                   const int             width=1,           // largura da linha fan
                   const bool            back=false,        // no fundo
                   const bool            selection=true,    // destaque para mover
                   const bool            hidden=true,       // ocultar na lista de obj
                   const long            z_order=0)         // prioridade para clique
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar Fibonacci Fan pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha para criar \" Fibonacci Fan\"! Código de erro = ",GetLastError())
      return(false);
     }
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir estilo de linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover o fan
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,         // número de linhas dos níveis
                      double          &values[],      // valores de linhas dos níveis
                      color           &colors[],      // cor das linhas dos níveis
                      ENUM_LINE_STYLE &styles[],      // estilo das linhas dos níveis
                      int             &widths[],      // largura das linhas dos níveis
                      const long      chart_ID=0,     // ID do gráfico
                      const string    name="FiboFan") // nome do fan
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem da Fibonacci Fan                        |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // ID do gráfico
                        const string name="FiboFan", // nome do fan
                        const int    point_index=0,  // índice do ponto de ancoragem
                        datetime     time=0,         // coordenada do ponto de ancorag
```

```
                        double       price=0)        // coordenada do ponto de ancorac
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir nome da Fibo                                             |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // ID do gráfico
                   const string name="FiboFan") // nome do fan
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir Gann Fan
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \" Fibonacci Fan\"! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem Fibonacci Fan e              |
//| definir os valores padrões estabelecidos para os pontos vazio    |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
```

```
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 200 pontos a mais do que o
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem da Fibonacci Fan
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
```

© 2000-2024, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a Fibonacci Fan
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar um objeto
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do fan
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1<accuracy-1)
         p1+=1;
      //--- mover o ponto
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars/4;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      //--- mover o ponto
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
```

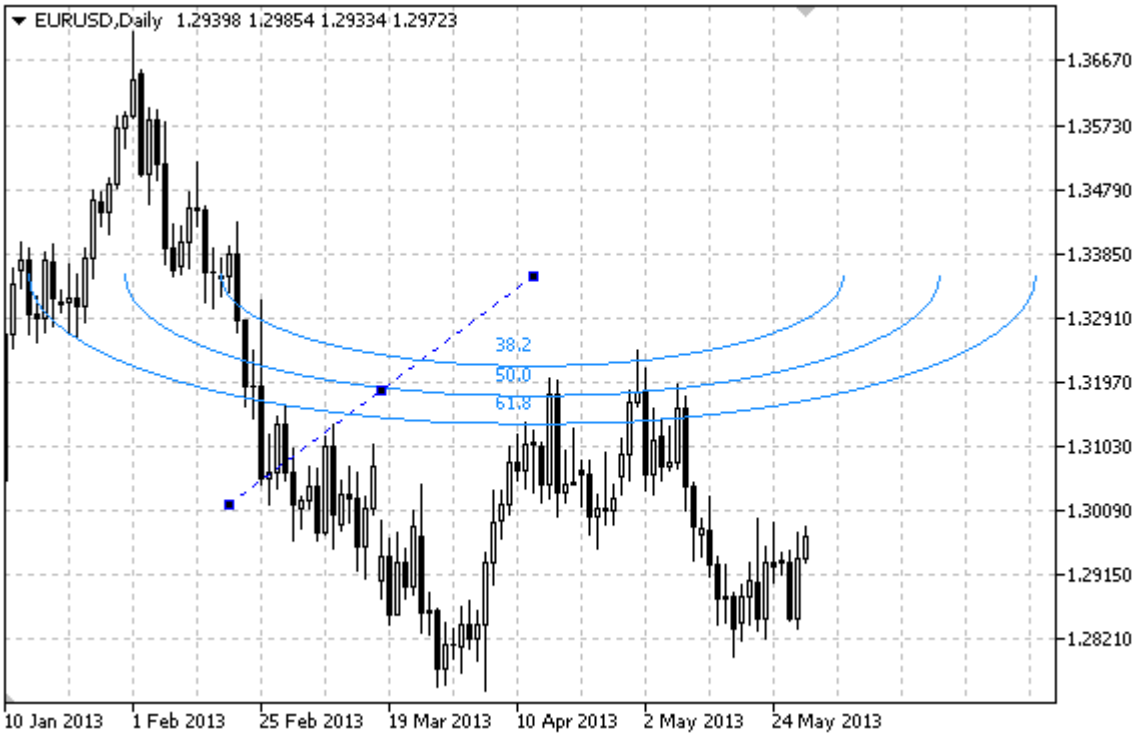
© 2000-2024, MetaQuotes Ltd.

```
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOARC

Arcos Fibonacci.



Observação

Para "Arcos Fibonacci Arcs", é possível determinar o modo de exibição da elipse inteira. O raio de curvatura pode ser especificado, alterando a escala e as coordenadas dos pontos de ancoragem.

Você também pode especificar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move a Arcos Fibonacci no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Arcos Fibonacci\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboArc";          // Nome do objeto
input int             InpDate1=25;                // Data do 1° ponto, %
input int             InpPrice1=25;               // Preço do 1° ponto, %
input int             InpDate2=35;                // Data do 2° ponto, %
input int             InpPrice2=55;               // Preço do 2° ponto, %
input double          InpScale=3.0;               // Escala
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpFullEllipse=true;       // Formas dos arcos
input color           InpColor=clrRed;           // Cor da linha
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo da linha
input int             InpWidth=2;                // Largura da linha
input bool            InpBack=false;             // Objeto de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Arcos Fibonacci pelas coordenadas fornecidas               |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,         // ID do gráfico
                   const string          name="FiboArc",     // nome do objeto
                   const int             sub_window=0,       // índice da sub-janela
                   datetime              time1=0,            // primeiro ponto de tem
                   double                price1=0,           // primeiro ponto de pre
                   datetime              time2=0,            // segundo ponto de tempo
                   double                price2=0,           // segundo ponto de preço
                   const double          scale=1.0,          // escala
                   const bool            full_ellipse=false, // forma dos arcos
                   const color           clr=clrRed,         // cor da linha
                   const ENUM_LINE_STYLE style=STYLE_SOLID,  // estilo da linha
                   const int             width=1,            // largura da linha
                   const bool            back=false,         // no fundo
                   const bool            selection=true,     // destacar para mover
                   const bool            hidden=true,        // ocultar na lista de ob
                   const long            z_order=0)          // prioridade para clica
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar Arcos Fibonacci pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Arcos Fibonacci\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir a escala
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- definir visualização dos arcos como uma elipse completa (true) ou a metade dela
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir estilo de linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover os arcos
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int             levels,         // número de linhas dos níveis
                      double          &values[],      // valores de linhas dos níveis
                      color           &colors[],      // cor das linhas dos níveis
                      ENUM_LINE_STYLE &styles[],      // estilo das linhas dos níveis
                      int             &widths[],      // largura das linhas dos níveis
                      const long      chart_ID=0,     // ID do gráfico
                      const string    name="FiboArc") // nome do objeto
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem dos Arcos Fibonacci                     |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,      // ID do gráfico
                        const string name="FiboArc", // nome do objeto
                        const int    point_index=0,  // índice do ponto de ancoragem
                        datetime     time=0,         // coordenada do ponto de ancorag
                        double       price=0)        // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Arcos Fibonacci                                          |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,      // ID do gráfico
                   const string name="FiboArc") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir\"Arcos Fibonacci\"! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem dos Arcos Fibonacci e definir|
```

© 2000-2024, MetaQuotes Ltd.

```
//| valores padrão para aqueles vazios                                |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto não está definido, está localizado 9 barras a esque
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que c
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem dos Arcos Fibonacc
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar Arcos Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar um objeto
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int v_steps=accuracy/5;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1<accuracy-1)
         p1+=1;
      //--- mover o ponto
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars/5;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2<bars-1)
         d2+=1;
      //--- mover o ponto
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

Canal Fibonacci.

Observação

Para "Canal Fibonacci", é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com as propriedades OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Você também pode especificar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move o Canal Fibonacci no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha um objeto gráfico tipo \"Canal Fibonacci\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboChannel";      // Nome do canal
input int             InpDate1=20;                // Data do 1° ponto, %
input int             InpPrice1=10;               // Preço do 1° ponto, %
input int             InpDate2=60;                // Data do 2° ponto, %
input int             InpPrice2=30;               // Preço do 2° ponto, %
input int             InpDate3=20;                // Data do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpPrice3=25;              // Preço do 3° ponto, %
input color           InpColor=clrRed;           // Cor do canal
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas do canal
input int             InpWidth=2;                // Largura das linhas do canal
input bool            InpBack=false;             // Canal de Fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpRayLeft=false;          // Continuação do Canal para a esquer
input bool            InpRayRight=false;         // Continuação do Canal para a direit
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Canal Fibonacci pelas coordenadas fornecidas               |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long            chart_ID=0,         // ID do gráfico
                       const string          name="FiboChannel", // nome do canal
                       const int             sub_window=0,       // índice da sub-jane
                       datetime              time1=0,            // primeiro ponto de
                       double                price1=0,           // primeiro ponto de
                       datetime              time2=0,            // segundo ponto de t
                       double                price2=0,           // segundo ponto de p
                       datetime              time3=0,            // terceiro ponto de
                       double                price3=0,           // terceiro ponto de
                       const color           clr=clrRed,         // cor do sinal
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // estilo das linhas
                       const int             width=1,            // largura das linhas
                       const bool            back=false,         // no fundo
                       const bool            selection=true,     // destacar para move
                       const bool            ray_left=false,     // continuação do car
                       const bool            ray_right=false,    // continuação do car
                       const bool            hidden=true,        // ocultar na lista d
                       const long            z_order=0)          // prioridade para cl
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um canal pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": falha ao criar \"Canal Fibonacci\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir a cor do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha do canal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do car
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,             // número de linhas dos
                          double          &values[],          // valores de linhas dos
                          color           &colors[],          // cor das linhas dos ní
                          ENUM_LINE_STYLE &styles[],          // estilo das linhas dos
                          int             &widths[],          // largura das linhas do
                          const long      chart_ID=0,         // ID do gráfico
                          const string    name="FiboChannel") // nome do objeto
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- largura de nível
```

© 2000-2024, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem do Canal Fibonacci                      |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,         // ID do gráfico
                            const string name="FiboChannel", // nome do canal
                            const int    point_index=0,      // índice do ponto de anc
                            datetime     time=0,             // coordenada do ponto de
                            double       price=0)            // coordenada do ponto de
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o canal                                                  |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,         // ID do gráfico
                       const string name="FiboChannel") // nome do canal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o canal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Canal Fibonacci\"! Código de erro = ",GetLastError()
      return(false);
     }
//--- sucesso na execução
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores dos pontos de ancoragem do Canal Fibonacci e         |
//| definir os valores padrões estabelecidos para os pontos vazio    |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- se o tempo do segundo ponto (a direita)não está definido, será na barra atual
   if(!time2)
      time2=TimeCurrent();
//--- se o preço do segundo ponto não está definido, ele terá valor Bid
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto (a esquerda) não está definido, está localizado 9 b
   if(!time1)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do terceiro ponto não está definido,ele coincide com o tempo do prime
   if(!time3)
      time3=time1;
//--- se o preço do terceiro ponto não está definido, é igual ao preço do segundo pont
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do canal
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o canal
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar Canal Fibonacci
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do canal
//--- contador de loop
   int h_steps=bars/10;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d1>1)
         d1-=1;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- mover o ponto
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int v_steps=accuracy/10;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2>1)
         p2-=1;
      //--- mover o ponto
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy/15;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p3<accuracy-1)
         p3+=1;
      //--- mover o ponto
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EXPANSION

Fibonacci Expansion.

Observação

Para "Expansão Fibonacci", é possível determinar o modo da continuação da sua exibição para a direita e/ou à esquerda (de acordo com a propriedade OBJPROP_RAY_RIGHT e OBJPROP_RAY_LEFT).

Você também pode especificar o número de linhas de níveis, os seus valores e cores.

Exemplo

O seguinte script cria e move a Expansão Fibonacci no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Expansão Fibonacci\"."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="FiboExpansion";   // Nome do objeto
input int             InpDate1=10;               // Data do 1° ponto, %
input int             InpPrice1=55;              // Preço do 1° ponto, %
input int             InpDate2=30;               // Data do 2° ponto, %
input int             InpPrice2=10;              // Preço do 2° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpDate3=80;               // Data do 3° ponto, %
input int             InpPrice3=75;              // Preço do 3° ponto, %
input color           InpColor=clrRed;           // Cor do objeto
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas
input int             InpWidth=2;                // Largura das linha
input bool            InpBack=false;             // Objeto de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpRayLeft=false;          // Continuação do objeto para a esque
input bool            InpRayRight=false;         // Continuação do objeto para a direi
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Extensão Fibonacci pelas coordenadas fornecidas            |
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long            chart_ID=0,           // ID de gráfico
                         const string          name="FiboExpansion", // nome do canal
                         const int             sub_window=0,         // índice da sub-
                         datetime              time1=0,              // primeiro ponto
                         double                price1=0,             // primeiro ponto
                         datetime              time2=0,              // segundo ponto
                         double                price2=0,             // segundo ponto
                         datetime              time3=0,              // terceiro ponto
                         double                price3=0,             // terceiro ponto
                         const color           clr=clrRed,           // cor do objeto
                         const ENUM_LINE_STYLE style=STYLE_SOLID,    // estilo das lin
                         const int             width=1,              // largura das li
                         const bool            back=false,           // no fundo
                         const bool            selection=true,       // destaque para
                         const bool            ray_left=false,       // continuação do
                         const bool            ray_right=false,      // continuação do
                         const bool            hidden=true,          // ocultar na lis
                         const long            z_order=0)            // prioridade par
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- Criar Extensão Fibonacci pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": falha para criar \"Extensão Fibonacci\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- definir cor do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do obj
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- habilitar (true) ou desabilitar (false) o modo de continuação da exibição do obj
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o número de níveis e seus parâmetros                     |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int             levels,                // número de linhas
                            double          &values[],             // valores de linhas
                            color           &colors[],             // cor das linhas d
                            ENUM_LINE_STYLE &styles[],             // estilo das linhas
                            int             &widths[],             // largura das linha
                            const long      chart_ID=0,            // ID do gráfico
                            const string    name="FiboExpansion") // nome do objeto
  {
//--- verificar tamanhos do array
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": comprimento do array não corresponder ao número de níveis,
      return(false);
     }
//--- definir o número de níveis
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- definir as propriedades de níveis no loop
   for(int i=0;i<levels;i++)
     {
      //--- valor do nível
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- cor do nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- estilo de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- largura de nível
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- descrição de nível
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem da Expansão Fibonacci                   |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID do gráfico
                              const string name="FiboExpansion", // nome do objeto
                              const int    point_index=0,        // índice do ponto de
                              datetime     time=0,               // coordenada do pont
                              double       price=0)              // coordenada do pont
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Expansão Fibonacci                                       |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID do gráfico
                         const string name="FiboExpansion") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \"Expansão Fibonacci\"! Código de erro = ",GetLastErro
      return(false);
     }
```

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Vê valores dos pontos de ancoragem da Expansão Fibonacci e       |
//| definir os valores padrões estabelecidos para os pontos vazio    |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- se o tempo do terceiro ponto (a direita)não está definido, será na barra atual
   if(!time3)
      time3=TimeCurrent();
//--- se o preço do terceiro ponto não está definido, ele terá valor Bid
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto (a esquerda) não está definido, ele está localizado
//--- array para receber o tempo de abertura das últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- definir o primeiro ponto 9 barras a esquerda do segundo
      time1=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, é igual ao terceiro ponto
   if(!price1)
      price1=price3;
//--- se o tempo do segundo ponto não está definido, está localizado a 7 barras deixado
   if(!time2)
      time2=temp[2];
//--- se o preço do primeiro ponto não está definido, mover 250 pontos a mais do que o
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar Expansão Fibonacci
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar Expansão Fibonacci
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int v_steps=accuracy/10;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
```

© 2000-2024, MetaQuotes Ltd.

```
      if(p1>1)
         p1-=1;
      //--- mover o ponto
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy/2;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p3>1)
         p3-=1;
      //--- mover o ponto
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy*4/5;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p2<accuracy-1)
         p2+=1;
      //--- mover o ponto
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

Onda Móvel Elliott.

Observação

Para "Onda Móvel Elliott", é possível habilitar/desabilitar o modo de pontos de conexão das linhas (propriedade OBJPROP_DRAWLINES), bem como definir o nível de posicionamento da onda (a partir da enumeração ENUM_ELLIOT_WAVE_DEGREE).

Exemplo

[parcial]O seguinte script cria e move onda móvel Elliott no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha \"Onda Móvel Elliott\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string                    InpName="ElliotWave5";    // Nome do objeto
input int                       InpDate1=10;              // Data do 1° ponto, %
input int                       InpPrice1=90;             // Preço do 1° ponto, %
input int                       InpDate2=20;              // Data do 2° ponto, %
input int                       InpPrice2=40;             // Preço do 2° ponto, %
input int                       InpDate3=30;              // Data do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                     InpPrice3=60;            // Preço do 3° ponto, %
input int                     InpDate4=40;             // Data do 4° ponto, %
input int                     InpPrice4=10;            // Preço do 4° ponto, %
input int                     InpDate5=60;             // Data do 5° ponto, %
input int                     InpPrice5=40;            // Preço do 5° ponto, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Nível
input bool                    InpDrawLines=true;       // Exibindo as linhas
input color                   InpColor=clrRed;         // Cor das linhas
input ENUM_LINE_STYLE         InpStyle=STYLE_DASH;     // Estilo das linhas
input int                     InpWidth=2;              // Largura das linha
input bool                    InpBack=false;           // Objeto de fundo
input bool                    InpSelection=true;       // Destaque para mover
input bool                    InpHidden=true;          // Ocultar na lista de objetos
input long                    InpZOrder=0;             // Prioridade para clique do mo
//+------------------------------------------------------------------+
//| Criar "Onda Móvel Elliott" pelas coordenadas fornecidas          |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                    chart_ID=0,              // ID do
                       const string                  name="ElliotWave3",      // nome
                       const int                     sub_window=0,            // índic
                       datetime                      time1=0,                 // prime
                       double                        price1=0,                //primei
                       datetime                      time2=0,                 // segun
                       double                        price2=0,                // segun
                       datetime                      time3=0,                 // terce
                       double                        price3=0,                // terce
                       datetime                      time4=0,                 // quart
                       double                        price4=0,                // quart
                       datetime                      time5=0,                 // quint
                       double                        price5=0,                // quint
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // grau
                       const bool                    draw_lines=true,         // exibi
                       const color                   clr=clrRed,              // cor d
                       const ENUM_LINE_STYLE         style=STYLE_SOLID,       // estil
                       const int                     width=1,                 // largu
                       const bool                    back=false,              // no fu
                       const bool                    selection=true,          // desta
                       const bool                    hidden=true,             // oculta
                       const long                    z_order=0)               // prior
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- redefine o valor de erro
   ResetLastError();
//--- Criar "Onda Móvel Elliott" pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

```
            ": falha ao criar \""Onda Móvel Elliott"\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- definir grau (tamanho da onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- habilitar (true) ou desabilitar (false) o modo de exibição das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- definir cor do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Move o ponto de ancoragem da Onda Móvel Elliott                  |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,         // ID do gráfico
                            const string name="ElliotWave5", // nome do objeto
                            const int    point_index=0,      // índice do ponto de anc
                            datetime     time=0,             // coordenada do ponto de
                            double       price=0)            // coordenada do ponto de
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
```

```
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Onda Móvel Elliott                                       |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // ID do gráfico
                       const string name="ElliotWave5") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \""Onda Móvel Elliott"\"! Código de erro = ",GetLastEr
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de pontos de ancoragem da Onda Móvel Elliott   |
//| definir os valores padrões estabelecidos para aqueles vazios     |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- array para receber o tempo de abertura das últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
//--- receber dados
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- receber o valor de um ponto no gráfico atual
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do primeiro ponto não está definido, serão 9 barras a esquerda a part
   if(!time1)
      time1=temp[0];
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto não está definido, serão 7 barras a esquerda a part
   if(!time2)
      time2=temp[2];
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price2)
      price2=price1-300*point;
//--- se o tempo do primeiro ponto não está definido, serão 5 barras a esquerda a part
   if(!time3)
      time3=temp[4];
//--- se o preço do primeiro ponto não está definido, mover 250 pontos a mais do que o
   if(!price3)
      price3=price1-250*point;
//--- se o tempo do primeiro ponto não está definido, serão 3 barras a esquerda a part
   if(!time4)
      time4=temp[6];
//--- se o preço do primeiro ponto não está definido, mover 550 pontos a mais do que o
   if(!price4)
      price4=price1-550*point;
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time5)
      time5=temp[9];
//--- se o preço do primeiro ponto não está definido, mover 450 pontos a mais do que o
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
```

© 2000-2024, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a Onda Móvel Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- Criar Onda Móvel Elliott
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int v_steps=accuracy/5;
//--- mover o quinto ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p5<accuracy-1)
         p5+=1;
      //--- mover o ponto
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy/5;
//--- mover o segundo e terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar os seguintes valores
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- deslocar os pontos
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy*4/5;
//--- mover o primeiro e quarto ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar os seguintes valores
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- deslocar os pontos
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

Correção Onda Elliott.

Observação

Para "Correção Onda Elliott", é possível habilitar/desabilitar o modo de pontos de conexão das linhas (propriedade OBJPROP_DRAWLINES), bem como definir o nível de posicionamento da onda (a partir da enumeração ENUM_ELLIOT_WAVE_DEGREE).

Exemplo

O seguinte script cria e move a correção da onda Elliott no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Correção da Onda Elliott\"."
#property description "Coordenadas dos pontos de ancoragem são definidos em percentage
#property description "tamanho."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string                    InpName="ElliotWave3";    // Nome do objeto
input int                       InpDate1=10;              // Data do 1° ponto, %
input int                       InpPrice1=90;             // Preço do 1° ponto, %
input int                       InpDate2=30;              // Data do 2° ponto, %
input int                       InpPrice2=10;             // Preço do 2° ponto, %
input int                       InpPrice2=10;             // Data do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                     InpPrice3=40;            // Preço do 3° ponto, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Nível
input bool                    InpDrawLines=true;       // Exibindo as linhas
input color                   InpColor=clrRed;         // Cor das linhas
input ENUM_LINE_STYLE         InpStyle=STYLE_DASH;     // Estilo das linhas
input int                     InpWidth=2;              // Largura das linha
input bool                    InpBack=false;           // Objeto de fundo
input bool                    InpSelection=true;       // Destaque para mover
input bool                    InpHidden=true;          // Ocultar na lista de objetos
input long                    InpZOrder=0;             // Prioridade para clique do mo
//+------------------------------------------------------------------+
//| Criar "Correção da Onda Elliott" pelas coordenadas fornecidas    |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                    chart_ID=0,              // ID do
                       const string                  name="ElliotWave3",      // nome
                       const int                     sub_window=0,            // índic
                       datetime                      time1=0,                 // prime
                       double                        price1=0,                //prime
                       datetime                      time2=0,                 // segun
                       double                        price2=0,                // segun
                       datetime                      time3=0,                 // terce
                       double                        price3=0,                // terce
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // grau
                       const bool                    draw_lines=true,         // exibi
                       const color                   clr=clrRed,              // cor d
                       const ENUM_LINE_STYLE         style=STYLE_SOLID,       // estil
                       const int                     width=1,                 // largu
                       const bool                    back=false,              // no fu
                       const bool                    selection=true,          // desta
                       const bool                    hidden=true,             // ocult
                       const long                    z_order=0)               // prio
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- Criar "Correção da Onda Elliott" pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": falha ao criar \""Correção da Onda Elliott"\"! Código de erro = ",GetLa
      return(false);
     }
//--- definir grau (tamanho da onda)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- habilitar (true) ou desabilitar (false) o modo de exibição das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- definir cor do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir o estilo da linha
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do canal
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Move o ponto de ancoragem de Correção da Elliott                 |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,         // ID do gráfico
                            const string name="ElliotWave3", // nome do objeto
                            const int    point_index=0,      // índice do ponto de anc
                            datetime     time=0,             // coordenada do ponto de
                            double       price=0)            // coordenada do ponto de
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir Correção da Onda Elliott                                 |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,         // ID do gráfico
```

```
                        const string name="ElliotWave3") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir \""Correção da Onda Elliott"\"! Código de erro = ",Get
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de pontos de ancoragem da Correção Onda Elliott|
//| e os valores padrão estabelecidos para os pontos vazios          |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- array para receber o tempo de abertura das últimas 10 barras
   datetime temp[];
   ArrayResize(temp,10);
//--- receber dados
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- receber o valor de um ponto no gráfico atual
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do primeiro ponto não está definido, serão 9 barras a esquerda a part
   if(!time1)
      time1=temp[0];
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do primeiro ponto não está definido, serão 5 barras a esquerda a part
   if(!time2)
      time2=temp[4];
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price2)
      price2=price1-300*point;
//--- se o tempo do primeiro ponto não está definido, será uma barra a esquerda a part
   if(!time3)
      time3=temp[8];
//--- se o preço do primeiro ponto não está definido, mover 200 pontos a mais do que o
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Programa Script da função start (iniciar)                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar a Correção da Onda Elliott
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- Criar Correção da Onda Elliott
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidde
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem
//--- contador de loop
   int v_steps=accuracy/5;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p3<accuracy-1)
         p3+=1;
      //--- mover o ponto
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy*4/5;
//--- movimentar o primeiro e segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar os seguintes valores
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- deslocar os pontos
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_RECTANGLE

Retângulo.

Observação

Para retângulo, o modo de preenchimento com a cor pode ser definida usando a propriedade OBJPROP_FILL.

Exemplo

O seguinte script cria e move o retângulo no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria o retângulo no gráfico."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Rectangle"; // Nome do retângulo
input int             InpDate1=40;         // Data do 1° ponto, %
input int             InpPrice1=40;        // Preço do 1° ponto, %
input int             InpDate2=60;         // Data do 2° ponto, %
input int             InpPrice2=60;        // Preço do 2° ponto, %
input color           InpColor=clrRed;     // Cor do retângulo
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Estilo das linhas do retângulo
```

© 2000-2024, MetaQuotes Ltd.

```
input int             InpWidth=2;          // Largura das linhas do retângulo
input bool            InpFill=true;        // Preenchendo o retângulo com cor
input bool            InpBack=false;       // Retângulo de fundo
input bool            InpSelection=true;   // Destaque para mover
input bool            InpHidden=true;      // Ocultar na lista de objeto
input long            InpZOrder=0;         // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar retângulo pelas coordenadas fornecidas                     |
//+------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // ID do gráfico
                     const string          name="Rectangle",  // nome do retângulo
                     const int             sub_window=0,      // índice da sub-janela
                     datetime              time1=0,           // primeiro ponto de ter
                     double                price1=0,          // primeiro ponto de pre
                     datetime              time2=0,           // segundo ponto de temp
                     double                price2=0,          // segundo ponto de pre
                     const color           clr=clrRed,        // cor do retângulo
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas do
                     const int             width=1,           // largura das linhas d
                     const bool            fill=false,        // preenchimento do ret
                     const bool            back=false,        // no fundo
                     const bool            selection=true,    // destaque para mover
                     const bool            hidden=true,       // ocultar na lista de
                     const long            z_order=0)         // prioridade para cliq
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um retângulo pelas coordenadas dadas
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": falha ao criar um retângulo! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir a cor do retângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linhas do retângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas do retângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento do retângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover o retângul
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
```

```
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem do retângulo                            |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // ID do gráfico
                          const string name="Rectangle", // nome do retângulo
                          const int    point_index=0,    // índice do ponto de ancorag
                          datetime     time=0,           // coordenada do ponto de anc
                          double       price=0)          // coordenada do ponto de anc
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o retângulo                                              |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="Rectangle") // nome do retângulo
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o retângulo
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o retângulo! Código de erro = ",GetLastError());
```

```
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores pontos de ancoragem do retângulo e definir           |
//| valores padrão para aqueles vazios                               |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem do retângulo
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar o retângulo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- criar o retângulo
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem do retângulo
//--- contador de loop
   int h_steps=bars/2;
//--- mover os pontos de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar os seguintes valores
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- deslocar os pontos
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover os pontos de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar os seguintes valores
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- deslocar os pontos
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o retângulo do gráfico
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TRIANGLE

Triângulo.

Observação

Para triângulo, o modo de preenchimento com pode ser definido usando a propriedade OBJPROP_FILL.

Exemplo

O seguinte script cria e movimenta o triângulo no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria o triângulo no gráfico."
#property description "Coordenadas de ponto de ancoragem são definidas em porcentagem
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Triangle";        // Nome do triângulo
input int             InpDate1=25;               // Data do 1° ponto, %
input int             InpPrice1=50;              // Preço do 1° ponto, %
input int             InpDate2=70;               // Data do 2° ponto, %
input int             InpPrice2=70;              // Preço do 2° ponto, %
input int             InpDate3=65;               // Data do 3° ponto, %
input int             InpPrice3=20;              // Preço do 3° ponto, %
```

© 2000-2024, MetaQuotes Ltd.

```
input color           InpColor=clrRed;           // Cor do triângulo
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo das linhas triangulo
input int             InpWidth=2;                // Largura das linhas do triângulo
input bool            InpFill=false;             // Preenchendo triângulos com cor
input bool            InpBack=false;             // Triângulo de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar triângulo pelas coordenadas determinadas                   |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // ID do gráfico
                    const string          name="Triangle",   // nome do triângulo
                    const int             sub_window=0,      // índice da sub-janela
                    datetime              time1=0,           // primeiro ponto de tem
                    double                price1=0,          // primeiro ponto de pre
                    datetime              time2=0,           // segundo ponto de tempo
                    double                price2=0,          // segundo ponto de preço
                    datetime              time3=0,           // terceiro ponto de tem
                    double                price3=0,          // terceiro ponto de pre
                    const color           clr=clrRed,        // cor do triângulo
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas do
                    const int             width=1,           // largura das linhas do
                    const bool            fill=false,        // preenchendo triângulo
                    const bool            back=false,        // no fundo
                    const bool            selection=true,    // destaque para mover
                    const bool            hidden=true,       // ocultar na lista de o
                    const long            z_order=0)         // prioridade para o cli
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- criar triângulo pelas coordenadas determinadas
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": falha ao criar triângulo! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir a cor do triângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo das linhas do triângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da linha dos triângulos
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preencher o triângulo
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de destacar o triângulo para move
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover a ponto de ancoragem do triângulo                          |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // ID do grafico
                         const string name="Triangle", // nome do triângulo
                         const int    point_index=0,   // índice do ponto de ancoragem
                         datetime     time=0,          // coordenada do ponto de ancor
                         double       price=0)         // coordenada do ponto de ancor
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o triângulo                                              |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // ID do gráfico
                    const string name="Triangle") // nome do triângulo
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o triângulo
```

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao deletar o triângulo! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Confira valores dos pontos de ancoragem do triângulo             |
//| e definir padrão valores padrão para aqueles vazios              |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do terceiro ponto não está definido,ele coincide com o tempo do segun
   if(!time3)
      time3=time2;
//--- se o preço do terceiro ponto não está definido, é igual aquele do primeiro ponto
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
```

```
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas dos pontos de ancoragem do triângulo
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o triângulo
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar um triângulo
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem da triângulo
```

© 2000-2024, MetaQuotes Ltd.

```
//--- contador de loop
   int v_steps=accuracy*3/10;
//--- mover o primeiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p1>1)
         p1-=1;
      //--- mover o ponto
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars*9/20-1;
//--- mover o segundo ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d2>1)
         d2-=1;
      //--- mover o ponto
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   v_steps=accuracy/4;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p3<accuracy-1)
         p3+=1;
      //--- mover o ponto
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir triângulo do gráfico
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ELLIPSE

Elipse.

Observação

Para elipse, o modo de preenchimento com a cor pode ser definida usando a propriedade OBJPROP_FILL.

Exemplo

O seguinte script cria e move a Elipse no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria a Elipse no gráfico."
#property description "Coordenadas dos pontos de ancoragem são definidas"
#property description "em porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Ellipse";          // Nome da elipse
input int             InpDate1=30;                // Data do 1° ponto, %
input int             InpPrice1=20;               // Preço do 1° ponto, %
input int             InpDate2=70;                // Data do 2° ponto, %
input int             InpPrice2=80;               // Preço do 2° ponto, %
input int             InpDate3=50;                // Data do 3° ponto, %
input int             InpPrice3=60;               // Preço do 3° ponto, %
input color           InpColor=clrRed;            // Cor da elipse
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Estilo de linhas da elipse
input int             InpWidth=2;                // Largura das linhas da elipse
input bool            InpFill=false;             // Preenchendo elipse com cor
input bool            InpBack=false;             // Elipse de fundo
input bool            InpSelection=true;         // Destaque para mover
input bool            InpHidden=true;            // Ocultar na lista de objeto
input long            InpZOrder=0;               // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar uma elipse pelas coordenadas fornecidas                    |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // ID de gráfico
                   const string          name="Ellipse",    // nome da elipse
                   const int             sub_window=0,      // índice da sub-janela
                   datetime              time1=0,           // primeiro ponto de tempo
                   double                price1=0,          // primeiro ponto de preço
                   datetime              time2=0,           // segundo ponto de tempo
                   double                price2=0,          // segundo ponto de preço
                   datetime              time3=0,           // terceiro ponto de tempo
                   double                price3=0,          // terceiro ponto de preço
                   const color           clr=clrRed,        // cor da elipse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo das linhas da el
                   const int             width=1,           // largura das linhas da e
                   const bool            fill=false,        // preenchimento da elipse
                   const bool            back=false,        // no fundo
                   const bool            selection=true,    // destaque para mover
                   const bool            hidden=true,       // ocultar na lista de obj
                   const long            z_order=0)         // prioridade para clique
  {
//--- definir coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma elipse pelas coordenadas fornecidas
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": falha ao criar uma elipse! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir uma cor de elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linhas da elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura das linhas da elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- habilitar (true) ou desabilitar (false) o modo de preenchimento da elipse
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- habilitar (true) ou desabilitar (false) o modo de destaque para mover a elipse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem da elipse                               |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // ID do gráfico
                        const string name="Ellipse", // nome da elipse
                        const int    point_index=0,  // índice do ponto de ancoragem
                        datetime     time=0,         // coordenada do ponto de ancorag
                        double       price=0)        // coordenada do ponto de ancorag
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir elipse                                                   |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // ID do gráfico
                   const string name="Ellipse") // nome da elipse
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir uma elipse
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": falha ao excluir uma elipse! Código de erro = ",GetLastError());
       return(false);
      }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Ver valores de pontos âncora da elipse e definir valores padrões |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- se o tempo do primeiro ponto não está definido, será na barra atual
   if(!time1)
      time1=TimeCurrent();
//--- se o preço do primeiro ponto não está definido, ele terá valor Bid
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- se o tempo do segundo ponto não está definido, está localizado a 9 barras deixad
   if(!time2)
     {
      //--- array para receber o tempo de abertura das últimos 10 barras
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- definir o segundo ponto 9 barras a esquerda do primeiro
      time2=temp[0];
     }
//--- se o preço do primeiro ponto não está definido, mover 300 pontos a mais do que o
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- se o tempo do terceiro ponto não está definido,ele coincide com o tempo do segun
   if(!time3)
      time3=time2;
//--- se o preço do terceiro ponto não está definido, é igual aquele do primeiro ponto
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
```

© 2000-2024, MetaQuotes Ltd.

```
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar as coordenadas de pontos de ancoragem da elipse
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar a elipse
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- criar uma elipse
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover os pontos de ancoragem da elipse
//--- contador de loop
```

© 2000-2024, MetaQuotes Ltd.

```
   int v_steps=accuracy/5;
//--- movimentar o primeiro e segundo ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar os seguintes valores
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- deslocar os pontos
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int h_steps=bars/5;
//--- mover o terceiro ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d3>1)
         d3-=1;
      //--- mover o ponto
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir elipse a partir do gráfico
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

Sinal Thumbs Up.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O seguinte script cria e move sinal Thumbs Up no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal \"Thumbs Up\"."
#property description "Coordenadas do ponto de ancoragem é configurado em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ThumbUp";     // Nome do sinal
input int               InpDate=75;            // Ponto de ancoragem da data em %
input int               InpPrice=25;           // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;  // Tipo de ancoragem
input color             InpColor=clrRed;       // Cor do sinal
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo da linha da borda
input int               InpWidth=5;           // Tamanho do sinal
input bool              InpBack=false;        // Sinal do fundo
input bool              InpSelection=true;    // Destaque para mover
input bool              InpHidden=true;       // Ocultar na lista de objetos
input long              InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar sinal Thumbs Up                                            |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long               chart_ID=0,           // ID do gráfic
                        const string             name="ThumbUp",       // nome do sinal
                        const int                sub_window=0,         // índice da sub
                        datetime                 time=0,               // ponto de anco
                        double                   price=0,              // ponto de anco
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo de ancor
                        const color              clr=clrRed,           // cor do sinal
                        const ENUM_LINE_STYLE    style=STYLE_SOLID,    // estilo de lin
                        const int                width=3,              // tamanho do si
                        const bool               back=false,           // no fundo
                        const bool               selection=true,       // destaque para
                        const bool               hidden=true,          // ocultar na li
                        const long               z_order=0)            // prioridade pa
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar sinal \"Thumbs Up\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // ID do gráfico
                      const string name="ThumbUp", // nome do objeto
                      datetime     time=0,         // coordenada do ponto de ancoragem
                      double       price=0)        // coordenada do ponto de ancoragem
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tipo de ancoragem do sinal Thumbs Up                     |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // ID do Grá
                              const string            name="ThumbUp",    // nome do ob
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de an
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastError
      return(false);
     }
```

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Thumbs Up                                          |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // ID do gráfico
                        const string name="ThumbUp") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Thumbs Up\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar sinal Thumbs Up no gráfico
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int h_steps=bars/4;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d>1)
         d-=1;
      //--- mover o ponto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int v_steps=accuracy/4;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p<accuracy-1)
         p+=1;
      //--- mover o ponto
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

Sinal Thumbs Down.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O seguinte script cria e move o sinal Thumbs Down no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal \"Thumbs Down\"."
#property description "Coordenadas do ponto de ancoragem é configurado em porcentagem
#property description "tamanho da janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ThumbDown";      // nome do sinal
input int               InpDate=25;               // Ponto de ancoragem da data em %
input int               InpPrice=75;              // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Tipo de ancoragem
```

© 2000-2024, MetaQuotes Ltd.

```
input color             InpColor=clrRed;         // Cor do sinal
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Estilo de linha da borda
input int               InpWidth=5;              // Tamanho do Sinal
input bool              InpBack=false;           // Fundo do sinal
input bool              InpSelection=true;       // Destaque para mover
input bool              InpHidden=true;          // Ocultar na lista de objetos
input long              InpZOrder=0;             // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar sinal Thumbs Down                                          |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // ID do gráf
                          const string            name="ThumbDown",     // nome do si
                          const int               sub_window=0,         // índice de s
                          datetime                time=0,               // ponto de a
                          double                  price=0,              // ponto de a
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo de an
                          const color             clr=clrRed,           // cor do sin
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de
                          const int               width=3,              // tamanho do
                          const bool              back=false,           // no fundo
                          const bool              selection=true,       // destaque p
                          const bool              hidden=true,          // ocultar na
                          const long              z_order=0)            // prioridade
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar sinal \"Thumbs Down\"! Código de erro = ",GetLastError()
      return(false);
     }
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // ID do gráfico
                        const string name="ThumbDown", // nome do objeto
                        datetime     time=0,           // coordenada do ponto de ancor
                        double       price=0)          // coordenada do ponto de ancor
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tipo de ancoragem do sinal Thumbs Down                   |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long               chart_ID=0,        // ID do gr
                                const string             name="ThumbDown",  // nome do
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastError
      return(false);
```

```
      }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Thumbs Down                                        |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // ID do gráfico
                          const string name="ThumbDown") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Thumbs Down\"! Código de erro = ",GetLastEr
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
```

© 2000-2024, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar sinal Thumbs Down no gráfico
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int h_steps=bars/4;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- contador de loop
   int v_steps=accuracy/4;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p>1)
         p-=1;
      //--- mover o ponto
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_UP

Sinal Arrow Up.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O seguinte script cria e move sinal Arrow Up no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal \"Arrow Up\"."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ArrowUp";     // Nome do sinal
input int               InpDate=25;            // Ponto de ancoragem da data em %
input int               InpPrice=25;           // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;  // Tipo de ancoragem
input color             InpColor=clrRed;       // Cor do sinal
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo da linha da borda
input int               InpWidth=5;           // Tamanho do sinal
input bool              InpBack=false;        // Sinal do fundo
input bool              InpSelection=false;   // Destaque para mover
input bool              InpHidden=true;       // Ocultar na lista de objetos
input long              InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar sinal Arrow Up                                             |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // ID do gráfico
                   const string            name="ArrowUp",       // nome do sinal
                   const int               sub_window=0,         // índice da sub-jane
                   datetime                time=0,               // ponto de ancorager
                   double                  price=0,              // ponto de ancorager
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo de ancoragem
                   const color             clr=clrRed,           // cor do sinal
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo de linha da
                   const int               width=3,              // tamanho do sinal
                   const bool              back=false,           // no fundo
                   const bool              selection=true,       // destaque para move
                   const bool              hidden=true,          // ocultar na lista d
                   const long              z_order=0)            // prioridade para cl
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar o sinal \"Arrow Up\"! Código de erro = ",GetLastError())
      return(false);
     }
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,      // ID do gráfico
                 const string name="ArrowUp", // nome do objeto
                 datetime     time=0,         // coordenada do ponto de ancoragem de t
                 double       price=0)        // coordenada do ponto de ancoragem de p
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tipo de ancoragem do sinal Arrow Down                  |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // ID do gráfico
                         const string            name="ArrowUp",    // nome do objeto
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de ancorag
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a ancoragem do ponto de locação
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastErro
      return(false);
     }
```

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Arrow Up                                           |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // ID do gráfico
                   const string name="ArrowUp") // nome sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Arrow Up\"! Código de erro = ",GetLastError
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar sinal Arrow Up no gráfico
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p<accuracy-1)
         p+=1;
      //--- mover o ponto
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_DOWN

Sinal Arrow Down.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O seguinte script cria e move sinal Arrow Down no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal \"Arrow Down\"."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ArrowDown";      // Nome do Sinal
input int               InpDate=75;               // Ponto de ancoragem da data em %
input int               InpPrice=75;              // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM;  // Tipo de Ancoragem
input color             InpColor=clrRed;          // Cor do sinal
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // Estilo de linha da borda
```

© 2000-2024, MetaQuotes Ltd.

```
input int                InpWidth=5;              // Tamanho do Sinal
input bool               InpBack=false;           // Fundo do sinal
input bool               InpSelection=false;      // Destaque para mover
input bool               InpHidden=true;          // Ocultar na lista de objetos
input long               InpZOrder=0;             // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar sinal Arrow Down                                           |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,            // ID do gráfico
                     const string            name="ArrowDown",      // nome do sinal
                     const int               sub_window=0,          // índice da sub-ja
                     datetime                time=0,                // ponto de ancorag
                     double                  price=0,               // ponto de ancorag
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM,  // tipo de ancorage
                     const color             clr=clrRed,            // cor do sinal
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,     // estilo da linha
                     const int               width=3,               // tamanho do sinal
                     const bool              back=false,            // no fundo
                     const bool              selection=true,        // destaque para mo
                     const bool              hidden=true,           // Ocultar na lista
                     const long              z_order=0)             // prioridade para
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar o sinal \"Arrow Down\"! Código de erro = ",GetLastError
      return(false);
     }
//--- tipo de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // ID do gráfico
                   const string name="ArrowDown", // nome do objeto
                   datetime     time=0,           // coordenada do ponto de ancoragem
                   double       price=0)          // coordenada do ponto de ancoragem
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tipo de ancoragem do sinal Arrow Down                  |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,        // ID do gráfic
                           const string            name="ArrowDown",  // nome do objet
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de anco
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar a ancoragem do ponto de locação
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Arrow Down                                         |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="ArrowDown") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal\"Arrow Down\"! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
```

```
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar o sinal Arrow Down no gráfico
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int v_steps=accuracy/2;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p>1)
         p-=1;
      //--- mover o ponto
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- redesenhar o gráfico
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_STOP

Sinal Stop.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O seguinte script cria e move sinal Stop no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal\"Stop\"."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ArrowStop";      // Nome do sinal
input int               InpDate=10;               // Ponto de ancoragem da data em %
input int               InpPrice=50;              // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM;  // Tipo de ancoragem
input color             InpColor=clrRed;          // Cor do sinal
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Estilo de linha da borda
input int               InpWidth=5;              // Tamanho do Sinal
input bool              InpBack=false;           // Fundo do sinal
input bool              InpSelection=false;      // Destaque para mover
input bool              InpHidden=true;          // Ocultar na lista de objetos
input long              InpZOrder=0;             // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar sinal Stop                                                 |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // ID do gráfico
                     const string            name="ArrowStop",     // nome do sinal
                     const int               sub_window=0,         // índice da sub-ja
                     datetime                time=0,               // ponto de ancorag
                     double                  price=0,              // ponto de ancorag
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo de ancorage
                     const color             clr=clrRed,           // cor do sinal
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo da linha
                     const int               width=3,              // tamanho do sinal
                     const bool              back=false,           // no fundo
                     const bool              selection=true,       // destaque para mo
                     const bool              hidden=true,          // Ocultar na lista
                     const long              z_order=0)            // prioridade para
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar o sinal\"Stop\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,        // ID do gráfico
                   const string name="ArrowStop", // nome do objeto
                   datetime     time=0,           // coordenada do ponto de ancoragem
                   double       price=0)          // coordenada do ponto de ancoragem
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tipo de ancoragem do sinal Stop                          |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // ID do gráfic
                           const string            name="ArrowStop",  // nome do objet
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // posição do p
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastErro
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Stop                                               |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="ArrowStop") // nome da etiqueta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Stop\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar sinal Stop no gráfico
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int h_steps=bars*2/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.025 segundos de atraso
      Sleep(25);
     }
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- redesenhar o gráfico
   ChartRedraw();
//--- contador de loop
   h_steps=bars*2/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.025 segundos de atraso
      Sleep(25);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_CHECK

Check sign.

Observação

Posição do ponto de ancoragem relativa ao sinal pode ser selecionado a partir do enumerador ENUM_ARROW_ANCHOR.

Sinais de grande porte (mais de 5) somente serão criados, definindo corretamente o valor da propriedade OBJPROP_WIDTH ao escrever um código no MetaEditor.

Exemplo

O script seguinte cria e move o sinal Check no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha o sinal \"Check\"."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="ArrowCheck"; // Nome do sinal
input int               InpDate=10;           // Ponto de ancoragem da data em %
input int               InpPrice=50;          // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Tipo de ancoragem
input color             InpColor=clrRed;      // Cor do sinal
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // Estilo da linha da borda
input int               InpWidth=5;           // Tamanho do sinal
input bool              InpBack=false;        // Sinal do fundo
input bool              InpSelection=false;   // Destaque para mover
input bool              InpHidden=true;       // Ocultar na lista de objetos
input long              InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar sinal Check                                                |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long               chart_ID=0,           // ID do gráfico
                      const string             name="ArrowCheck",    // nome do sinal
                      const int                sub_window=0,         // índice da sub-j
                      datetime                 time=0,               // ponto de ancora
                      double                   price=0,              // ponto de ancora
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tipo de ancorag
                      const color              clr=clrRed,           // cor do sinal
                      const ENUM_LINE_STYLE    style=STYLE_SOLID,    // estilo a linha
                      const int                width=3,              // tamanho do sina
                      const bool               back=false,           // no fundo
                      const bool               selection=true,       // destaque para r
                      const bool               hidden=true,          // Ocultar na list
                      const long               z_order=0)            // prioridade para
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar o sinal\"Check\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- tamanho da definição do sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // ID do gráfico
                    const string name="ArrowCheck", // nome do objeto
                    datetime     time=0,            // coordenada do ponto de ancorage
                    double       price=0)           // coordenada do ponto de ancorage
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tipo de ancoragem Check                                |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // ID do gráfic
                            const string            name="ArrowCheck", // nome do obje
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de anco
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastError
      return(false);
     }
```

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir os sinal Check                                           |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // ID do gráfico
                      const string name="ArrowCheck") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Check\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos para desenhar o sinal
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar o sinal Check no gráfico
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem e alterar sua posição em relação ao sinal
//--- contador de loop
   int h_steps=bars*2/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.025 segundos de atraso
      Sleep(25);
     }
//--- alterar a localização do ponto de ancoragem relativa ao sinal
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- redesenhar o gráfico
   ChartRedraw();
//--- contador de loop
   h_steps=bars*2/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<h_steps;i++)
     {
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.025 segundos de atraso
      Sleep(25);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- deletar o sinal do gráfico
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

Etiqueta de Preço à Esquerda

Exemplo

O seguinte script cria e move a Etiqueta Preço Lado Esquerdo no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria a Etiqueta Preço Lado Esquerdo no gráfico."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="LeftPrice";  // Nome da Etiqueta Preço
input int               InpDate=100;          // Ponto de ancoragem da data em %
input int               InpPrice=10;          // Ponto de ancoragem do preço em %
input color             InpColor=clrRed;      // Cor da Etiqueta Preço
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Estilo de linha da borda
input int               InpWidth=2;           // Tamanho da Etiqueta Preço
input bool              InpBack=false;        // Fundo Etiqueta
input bool              InpSelection=true;    // Destaque para mover
input bool              InpHidden=true;       // Ocultar na lista de objetos
input long              InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar Etiqueta Preço Lado Esquerdo                               |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // ID do gráfico
                          const string          name="LeftPrice",  // nome da etiqueta
                          const int             sub_window=0,      // índice da sub-ja
                          datetime              time=0,            // ponto de ancorag
                          double                price=0,           // ponto de ancorag
                          const color           clr=clrRed,        // cor da etiqueta
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha
                          const int             width=1,           // tamanho da etiqu
                          const bool            back=false,        // no fundo
                          const bool            selection=true,    // destaque para mo
                          const bool            hidden=true,       // ocultar na lista
                          const long            z_order=0)         // prioridade para
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma etiqueta de preço
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar a etiqueta preço lado esquerdo! Código de erro = ",GetLa
      return(false);
     }
//--- definir a cor da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,        // ID do gráfico
                        const string name="LeftPrice", // nome da etiqueta
                        datetime     time=0,           // coordenada do ponto de ancor
                        double       price=0)          // coordenada do ponto de ancor
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir a etiqueta preço lado esquerdo do gráfico                |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,        // ID gráfico
                          const string name="LeftPrice") // nome etiqueta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir a etiqueta preço lado esquerda! Código de erro = ",Get
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para configuração e alteração das coordenadas do ponto de ancoragem da etiqueta
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos por desenhar a etiqueta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar etiqueta preço lado esquerdo no gráfico
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem
//--- contador de loop
   int v_steps=accuracy*4/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p<accuracy-1)
         p+=1;
      //--- mover o ponto
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

Etiqueta Preço Lado Direito.

Exemplo

O seguinte script cria e move a Etiqueta Preço Lado Direito no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria a Etiqueta Preço Lado Direito no gráfico."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="RightPrice"; // Nome da Etiqueta Preço
input int               InpDate=0;            // Ponto de ancoragem da data em %
input int               InpPrice=90;          // Ponto de ancoragem do preço em %
input color             InpColor=clrRed;      // Cor da Etiqueta Preço
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Estilo de linha da borda
input int               InpWidth=2;           // Tamanho da Etiqueta Preço
input bool              InpBack=false;        // Fundo Etiqueta
input bool              InpSelection=true;    // Destaque para mover
input bool              InpHidden=true;       // Ocultar na lista de objetos
input long              InpZOrder=0;          // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar a etiqueta preço lado direito                              |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // ID do gráfico
                           const string          name="RightPrice", // nome do preço d
                           const int             sub_window=0,      // índice da sub-j
                           datetime              time=0,            // ponto de ancora
                           double                price=0,           // ponto de ancora
                           const color           clr=clrRed,        // cor da etiqueta
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha
                           const int             width=1,           // tamanho da etic
                           const bool            back=false,        // no fundo
                           const bool            selection=true,    // destaque para m
                           const bool            hidden=true,       // ocultar na list
                           const long            z_order=0)         // prioridade para
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma etiqueta de preço
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar a etiqueta de preço lado direito! Código de erro = ",Get
      return(false);
     }
//--- definir a cor da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // ID do gráfico
                         const string name="RightPrice", // nome da etiqueta
                         datetime     time=0,            // coordenada do ponto de anc
                         double       price=0)           // coordenada do ponto de anc
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir a etiqueta de preço lado direito do gráfico              |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // ID Gráfico
                           const string name="RightPrice") // nome da etiqueta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir a etiqueta de preço lado direito! Código de erro = ",G
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para configuração e alteração das coordenadas do ponto de ancoragem da etiqueta
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir os pontos por desenhar a etiqueta
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- criar a etiqueta de preço lado direito
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o ponto de ancoragem
//--- contador de loop
   int v_steps=accuracy*4/5;
//--- mover o ponto de ancoragem
   for(int i=0;i<v_steps;i++)
     {
      //--- usar o seguinte valor
      if(p>1)
         p-=1;
      //--- mover o ponto
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta do gráfico
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_BUY

Sinal Buy.

Exemplo

O seguinte script cria e move o sinal Buy no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinais \"Buy\" na janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input color InpColor=C'3,95,172'; // Cores dos sinais
//+------------------------------------------------------------------+
//| Criar sinal Buy                                                  |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // ID de gráfico
                    const string          name="ArrowBuy",   // nome do sinal
                    const int             sub_window=0,      // índice da sub-janela
                    datetime              time=0,            // ponto de ancoragem do
                    double                price=0,           // ponto de ancoragem do
                    const color           clr=C'3,95,172',   // cor do sinal
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha (quanc
                    const int             width=1,           // tamanho da linha (quar
                    const bool            back=false,        // no fundo
                    const bool            selection=false,   // destaque para mover
```

© 2000-2024, MetaQuotes Ltd.

```
                    const bool            hidden=true,       // ocultar na lista de ob
                    const long            z_order=0)         // prioridade para o clic
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar sinal \"Buy\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir um estilo de linha (quando destacado)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir um tamanho de linha (quando destacado)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // ID do gráfico
                  const string name="ArrowBuy", // nome do objeto
                  datetime     time=0,          // coordenar o ponto de ancoragem do t
                  double       price=0)         // coordenar o ponto de ancoragem do p
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
      {
       Print(__FUNCTION__,
             ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
       return(false);
      }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Buy                                                |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,      // ID do gráfico
                    const string name="ArrowBuy") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal \"Buy\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para armazenar datas das barras visíveis
   double   low[];  // array para armazenar preços Baixos das barras visíveis
   double   high[]; // array para armazenar preços Altos das barras visíveis
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array dos preços Baixos
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Falha ao copiar os valores dos preços Baixos! Código de erro = ",GetLastE
      return;
     }
//--- preencher o array dos preços Altos
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Falha ao copiar os valores dos preços Altos! Código de erro = ",GetLastEr
      return;
     }
//--- criar sinais Buy de ponto de baixa para cada barra visível
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- mover sinal Buy de ponto alto para cada barra visível
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- excluir sinais Buy
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW_SELL

Sell sign.

Exemplo

O seguinte script cria e move o sinal Sell no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha sinal \"Sell\" na janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input color InpColor=C'225,68,29'; // Cor do sinal
//+------------------------------------------------------------------+
//| Criar sinal Sell                                                 |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // ID do gráfico
                     const string          name="ArrowSell",  // nome do sinal
                     const int             sub_window=0,      // índice da sub-janela
                     datetime              time=0,            // ponto de ancoragem d
                     double                price=0,           // ponto de ancoragem d
                     const color           clr=C'225,68,29',  // cor do sinal
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha (quar
                     const int             width=1,           // tamanho da linha (qua
                     const bool            back=false,        // no fundo
                     const bool            selection=false,   // destaque para mover
```

© 2000-2024, MetaQuotes Ltd.

```
                      const bool            hidden=true,       // ocultar na lista de c
                      const long            z_order=0)         // prioridade para cliqu
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar o sinal
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar sinal \"Sell\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir uma cor de sinal
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir um estilo de linha (quando destacado)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir um tamanho de linha (quando destacado)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do sinal com o mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,        // ID do gráfico
                   const string name="ArrowSell",  // nome do objeto
                   datetime     time=0,            // coordenada do ponto de ancoragem
                   double       price=0)           // coordenada do ponto de ancoragem
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
      {
       Print(__FUNCTION__,
             ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
       return(false);
      }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir sinal Sell                                               |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="ArrowSell") // nome do sinal
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o sinal
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o sinal\"Sell\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para armazenar datas das barras visíveis
   double   low[];  // array para armazenar preços Baixos das barras visíveis
   double   high[]; // array para armazenar preços Altos das barras visíveis
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array dos preços Baixos
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Falha ao copiar os valores dos preços Baixos! Código de erro = ",GetLastE
      return;
     }
//--- preencher o array dos preços Altos
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("Falha ao copiar os valores dos preços Altos! Código de erro = ",GetLastEr
      return;
     }
//--- criar sinais Sell no ponto Alto para cada barra visível
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- mover sinais Sell para ponto Baixo em cada barra visível
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- excluir sinal Sell
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_ARROW

Objeto Seta.

Observação

Posição do ponto de ancoragem relativo ao objeto será selecionado a partir do ENUM_ARROW_ANCHOR.

As setas grandes (mais do que 5) somente serão criadas pela configuração de valores apropriados da propriedade OBJPROP_WIDTH quando escrever um código no MetaEditor.

O tipo de seta necessário podem ser selecionado, definindo um dos códigos dos símbolos da fonte Wingdings.

Exemplo

O seguinte script cria um objeto Seta no gráfico e altera seu tipo. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria uma seta aleatória na janela do gráfico."
#property description "Coordenadas do ponto de ancoragem é definido em"
#property description "porcentagem do tamanho da janela de gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="Arrow";        // Nome da seta
input int               InpDate=50;             // Ponto de ancoragem da data em %
```

© 2000-2024, MetaQuotes Ltd.

```
input int                InpPrice=50;            // Ponto de ancoragem do preço em %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;    // Tipo de ancoragem
input color              InpColor=clrDodgerBlue; // Cor da seta
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;   // Estilo de linha da borda
input int                InpWidth=10;            // Tamanho da seta
input bool               InpBack=false;          // Seta do fundo
input bool               InpSelection=false;     // Destaque para mover
input bool               InpHidden=true;         // Ocultar na lista de objeto
input long               InpZOrder=0;            // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Criar a seta                                                     |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // ID do gráfico
                 const string            name="Arrow",         // nome da seta
                 const int               sub_window=0,         // índice da sub-janela
                 datetime                time=0,               // ponto de ancoragem c
                 double                  price=0,              // ponto de ancoragem c
                 const uchar             arrow_code=252,       // código da seta
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // posição do ponto de
                 const color             clr=clrRed,           // cor da seta
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // estilo da linha da b
                 const int               width=3,              // tamanho da seta
                 const bool              back=false,           // no fundo
                 const bool              selection=true,       // destaque para mover
                 const bool              hidden=true,          // ocultar na lista de
                 const long              z_order=0)            // prioridade para clic
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeArrowEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma seta
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma seta! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir o código da seta
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir a cor da seta
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho da seta
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (true) o modo do movimento da seta com o mouse
//--- ao criar um objeto gráfico usando a função ObjectCreate, o objeto não pode ser
//--- destacado e movimentado por padrão. Dentro deste método, o parâmetro de seleção
//--- é verdade por padrão, tornando possível destacar e mover o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // ID do gráfico
               const string name="Arrow", // nome do objeto
               datetime     time=0,       // coordenada do ponto de ancoragem de tempo
               double       price=0)      // coordenada do ponto de ancoragem de preço
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o código da seta                                         |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // ID do gráfico
                     const string name="Arrow", // nome do objeto
                     const uchar  code=252)     // código da seta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar o código da seta
```

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
      Print(__FUNCTION__,
            ": falha ao alterar o código da seta! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tipo de ancoragem                                        |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // ID do gráfico
                       const string            name="Arrow",      // nome do objeto
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tipo de ancoragem
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": falha para alterar o tipo de ancoragem! Código de erro = ",GetLastEror
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir uma seta                                                 |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // ID do gráfico
                 const string name="Arrow") // nome da seta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir uma seta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir uma seta! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- tamanho do array de preço
   int accuracy=1000;
//--- arrays para armazenar data e valores de preço para serem usados
//--- para definir e alterar sinal das coordenadas do ponto de ancoragem
   datetime date[];
   double   price[];
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array de preços
//--- encontrar os maiores e menores valores do gráfico
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- definir uma etapa de mudança de um preço e preencher o array
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- definir pontos para desenhar a seta
   int d=InpDate*(bars-1)/100;
```

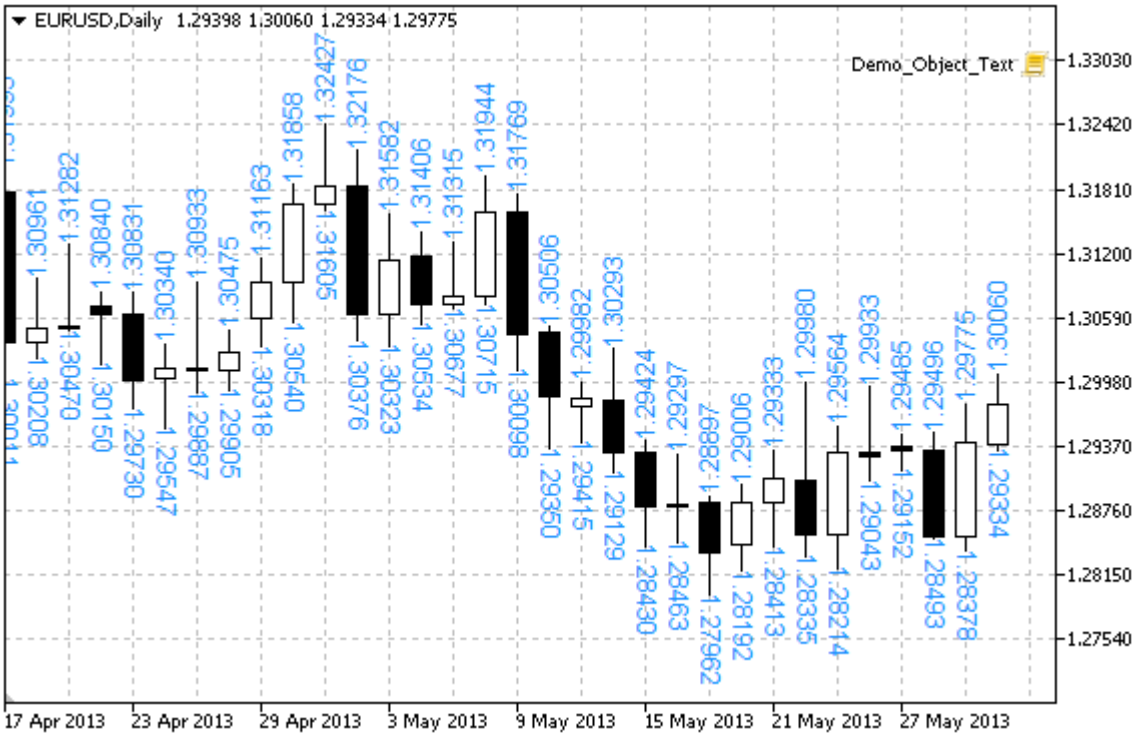
© 2000-2024, MetaQuotes Ltd.

```
   int p=InpPrice*(accuracy-1)/100;
//--- criar uma seta no gráfico
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico
   ChartRedraw();
//--- analisar todos os casos de criação de setas no loop
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // meio segundo de atraso
      Sleep(500);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a seta do gráfico
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_TEXT

Objeto Texto.



Observação

Posição de ponto de ancoragem em relação ao texto pode ser selecionado a partir do enumerador ENUM_ANCHOR_POINT. Você também pode alterar o ângulo de inclinação de texto usando a propriedade OBJPROP_ANGLE.

Exemplo

O seguinte o script cria vários objetos de texto no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria objeto gráfico de \"Texto\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpFont="Arial";          // Fonte
input int               InpFontSize=10;           // Tamanho da fonte
input color             InpColor=clrRed;          // Cor
input double            InpAngle=90.0;            // Ângulo de inclinação em graus
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // Tipo de ancoragem
input bool              InpBack=false;            // Objeto de fundo
input bool              InpSelection=false;       // Destaque para mover
input bool              InpHidden=true;           // Ocultar na lista de objetos
```

© 2000-2024, MetaQuotes Ltd.

```
input long              InpZOrder=0;             // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criando objeto Texto                                             |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,                // ID do gráfico
                const string            name="Text",               // nome do objeto
                const int               sub_window=0,              // índice da sub-jan
                datetime                time=0,                    // ponto de ancorage
                double                  price=0,                   // ponto de ancorage
                const string            text="Text",               // o próprio texto
                const string            font="Arial",              // fonte
                const int               font_size=10,              // tamanho da fonte
                const color             clr=clrRed,                // cor
                const double            angle=0.0,                 // inclinação do tex
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER,  // tipo de ancoragem
                const bool              back=false,                / no fundo
                const bool              selection=false,           // destaque para mov
                const bool              hidden=true,               // ocultar na lista
                const long              z_order=0)                 // prioridade para c
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeTextEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar objeto Texto
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar objeto \"Texto\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir o texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- definir o texto fonte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- definir tamanho da fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- definir o ângulo de inclinação do texto
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- habilitar (true) ou desabilitar (false) o modo de mover o objeto com o mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover ponto de ancoragem                                         |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // ID do gráfico
              const string name="Text", // nome do objeto
              datetime     time=0,      // coordenada do ponto de ancoragem do tempo
              double       price=0)     // coordenada do ponto de ancoragem do preço
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o texto do objeto                                        |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // ID do Gráfico
                const string name="Text", // nome do objeto
                const string text="Text") // texto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar texto do objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": falha ao alterar texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Excluir objeto Texto                                             |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // Id do Gráfico
                const string name="Text") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o objeto \"Texto\"! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // array para armazenar datas das barras visíveis
   double   low[];  // array para armazenar preços Baixos das barras visíveis
   double   high[]; // array para armazenar preços Altos das barras visíveis
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
       return;
      }
//--- preencher o array dos preços Baixos
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
      {
       Print("Falha ao copiar os valores dos preços Baixos! Código de erro = ",GetLastE
       return;
      }
//--- preencher o array dos preços Altos
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
      {
       Print("Falha ao copiar os valores dos preços Altos! Código de erro = ",GetLastEr
       return;
      }
//--- definir a forma como muitas vezes os textos serão exibidos
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definir o passo
   int step=1;
   switch(scale)
      {
       case 0:
          step=12;
          break;
       case 1:
          step=6;
          break;
       case 2:
          step=4;
          break;
       case 3:
          step=2;
          break;
      }
//--- criar textos para os valores de barras de Alta e de Baixa (com intervalos)
   for(int i=0;i<bars;i+=step)
      {
       //--- criar os textos
       if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
          InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
       if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
          InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
         {
          return;
         }
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- meio segundo de atraso
   Sleep(500);
//--- excluir os textos
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_LABEL

Objeto Etiqueta.

Observação

Posição do ponto de ancoragem relativa a etiqueta pode ser selecionada do enumerador ENUM_ANCHOR_POINT. Coordenadas do ponto de ancoragem são em pixels.

Você também pode selecionar uma ancoragem de etiqueta de texto no canto a partir do enumerador ENUM_BASE_CORNER.

Exemplo

O seguinte script cria e move o objeto Edit no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria objeto gráfico \"Etiqueta\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="Label";          // Nome etiqueta
input int               InpX=150;                 // distância do eixo X
input int               InpY=150;                 // distância do eixo Y
input string            InpFont="Arial";          // Fonte
input int               InpFontSize=14;           // Tamanho da fonte
input color             InpColor=clrRed;          // Cor
input double            InpAngle=0.0;             // Ângulo de inclinação em graus
```

© 2000-2024, MetaQuotes Ltd.

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Tipo de ancoragem
input bool              InpBack=false;           // Objeto de fundo
input bool              InpSelection=true;       // Destaque para mover
input bool              InpHidden=true;          // Ocultar na lista de objetos
input long              InpZOrder=0;             // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar um a etiqueta de texto                                     |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // ID do gráfico
                 const string            name="Label",             // nome da etiqueta
                 const int               sub_window=0,             // índice da sub-ja
                 const int               x=0,                      // coordenada X
                 const int               y=0,                      // coordenada Y
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // canto do gráfico
                 const string            text="Label",             // texto
                 const string            font="Arial",             // fonte
                 const int               font_size=10,             // tamanho da fonte
                 const color             clr=clrRed,               // cor
                 const double            angle=0.0,                // inclinação do te
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // tipo de ancorage
                 const bool              back=false,               // no fundo
                 const bool              selection=false,          // destaque para mo
                 const bool              hidden=true,              // ocultar na lista
                 const long              z_order=0)                // prioridade para
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar um a etiqueta de texto
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma etiqueta de texto! Código de erro = ",GetLastError()
      return(false);
     }
//--- definir coordenadas da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- determinar o canto do gráfico onde as coordenadas do ponto são definidas
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- definir o texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- definir o texto fonte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- definir tamanho da fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- definir o ângulo de inclinação do texto
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover a etiqueta de texto                                        |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // ID do gráfico
               const string name="Label", // nome da etiqueta
               const int    x=0,          // coordenada X
               const int    y=0)          // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover a etiqueta de texto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar canto do gráfico para vinculação da etiqueta             |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // ID do gráfi
                       const string           name="Label",             // nome da eti
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // canto do gr
  {
//--- redefine o valor de erro
   ResetLastError();
```

```
//--- alterar canto de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": falha ao alterar canto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o texto do objeto                                        |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // ID do gráfico
                     const string name="Label", // nome do objeto
                     const string text="Text")  // texto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar texto do objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": falha ao alterar texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Apagar uma etiqueta de texto                                     |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // ID do gráfico
                 const string name="Label") // nome da etiqueta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir a etiqueta de texto! Código de erro = ",GetLastError()
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- armazenar as coordenadas das etiquetas nas variáveis locais
   int x=InpX;
   int y=InpY;
//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
      return;
     }
//--- verificar a exatidão dos parâmetros de entrada
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- preparar texto inicial para a etiqueta
   string text;
   StringConcatenate(text,"Canto inferior direito: ",x,",",y);
//--- criar uma etiqueta de texto no gráfico
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(500);
//--- mover a etiqueta e alterar o seu texto simultaneamente
//--- número de iterações por eixos
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- mover a etiqueta para baixo
   for(int i=0;i<v_steps;i++)
     {
      //--- alterar a coordenada
      y+=2;
      //--- mover a etiqueta e alterar o seu texto
      MoveAndTextChange(x,y,"Canto superior esquerdo: ");
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- meio segundo de atraso
   Sleep(500);
//--- mover a etiqueta para a direita
   for(int i=0;i<h_steps;i++)
     {
      //--- alterar a coordenada
      x+=2;
      //--- mover a etiqueta e alterar o seu texto
      MoveAndTextChange(x,y,"Canto superior esquerdo: ");
     }
//--- meio segundo de atraso
   Sleep(500);
//--- mover a etiqueta para cima
   for(int i=0;i<v_steps;i++)
     {
      //--- alterar a coordenada
      y-=2;
      //--- mover a etiqueta e alterar o seu texto
      MoveAndTextChange(x,y,"Canto superior esquerdo: ");
     }
//--- meio segundo de atraso
   Sleep(500);
//--- mover a etiqueta para a esquerda
   for(int i=0;i<h_steps;i++)
     {
      //--- alterar a coordenada
      x-=2;
      //--- mover a etiqueta e alterar o seu texto
      MoveAndTextChange(x,y,"Canto superior esquerdo: ");
     }
//--- meio segundo de atraso
   Sleep(500);
//--- agora, mover o ponto para mudar o canto de ancoragem
//--- mover para o canto inferior esquerdo
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- alterar texto da etiqueta
   StringConcatenate(text,"Canto inferior direito: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redesenhar o gráfico e esperar por dois segundos
   ChartRedraw();
   Sleep(2000);
//--- mover para o canto inferior direito
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- alterar texto da etiqueta
   StringConcatenate(text,"Canto direito inferior: ",x,",",y);
```

© 2000-2024, MetaQuotes Ltd.

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redesenhar o gráfico e esperar por dois segundos
   ChartRedraw();
   Sleep(2000);
//--- mover para o canto superior direito
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- alterar texto da etiqueta
   StringConcatenate(text,"Canto inferior direito: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redesenhar o gráfico e esperar por dois segundos
   ChartRedraw();
   Sleep(2000);
//--- mover para o canto superior esquerdo
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- alterar texto da etiqueta
   StringConcatenate(text,"Canto inferior direito: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- redesenhar o gráfico e esperar por dois segundos
   ChartRedraw();
   Sleep(2000);
//--- excluir a etiqueta
   LabelDelete(0,InpName);
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| A função movimenta o objeto e altera o seu texto                 |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- mover a etiqueta
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- alterar texto da etiqueta
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- verificar se o funcionamento do script foi desativado a força
   if(IsStopped())
      return(false);
//--- redesenhar o gráfico
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
// 0.01 segundos de atraso
   Sleep(10);
//--- sair da função
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BUTTON

Objeto Botão.

Observação

Coordenadas do ponto de ancoragem são em pixels. Você pode selecionar o botão de ancoragem de canto a partir da ENUM_BASE_CORNER.

Exemplo

O seguinte script cria e move o objeto Botão no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria o Botão no gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="Button";             // Nome do Botão
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;  // Canto do gráfico para ancoragem
input string            InpFont="Arial";              // Fonte
input int               InpFontSize=14;               // Tamanho da fonte
input color             InpColor=clrBlack;            // Cor do texto
input color             InpBackColor=C'236,233,216';  // Fundo da cor
input color             InpBorderColor=clrNONE;       // Cor da Borda
input bool              InpState=false;               // Pressionado/Liberado
input bool              InpBack=false;                // Objeto de Fundo
input bool              InpSelection=false;           // Destaque para mover
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // Ocultar na lista de objeto
input long              InpZOrder=0;                 // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar o botão                                                    |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // ID do gráfico
                  const string            name="Button",            // nome do botão
                  const int               sub_window=0,             // índice da sub-j
                  const int               x=0,                      // coordenada X
                  const int               y=0,                      // coordenada Y
                  const int               width=50,                 // largura do botã
                  const int               height=18,                // altura do botão
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // canto do gráfic
                  const string            text="Button",            // texto
                  const string            font="Arial",             // fonte
                  const int               font_size=10,             // tamanho da font
                  const color             clr=clrBlack,             // cor do texto
                  const color             back_clr=C'236,233,216',  // cor do fundo
                  const color             border_clr=clrNONE,       // cor da borda
                  const bool              state=false,              // pressionada/lib
                  const bool              back=false,               // no fundo
                  const bool              selection=false,          // destaque para r
                  const bool              hidden=true,              // ocultar na list
                  const long              z_order=0)                // prioridade para
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar o botão
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar o botão! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir coordenadas do botão
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- definir tamanho do botão
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- determinar o canto do gráfico onde as coordenadas do ponto são definidas
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- definir o texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- definir o texto fonte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- definir tamanho da fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- definir a cor do texto
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a cor de fundo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- definir a cor da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- habilitar (true) ou desabilitar (false) o modo do movimento do botão com o mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover o botão                                                    |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // ID do gráfico
                const string name="Button", // nome do botão
                const int    x=0,           // coordenada X
                const int    y=0)           // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover o botão
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do botão! Código de erro = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada Y do botão! Código de erro = ",GetLastError
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tamanho do botão                                       |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // ID do gráfico
```

```
                        const string name="Button", // nome do botão
                        const int    width=50,      // largura do botão
                        const int    height=18)     // altura do botão
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar o tamanho do botão
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura do botão! Código de erro = ",GetLastError())
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura do botão! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar canto do gráfico para vinculação do botão                |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // ID do grá
                        const string           name="Button",            // nome do bo
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // canto do g
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar canto de ancoragem
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": falha ao alterar canto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar texto do botão                                           |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,     // ID do gráfico
                      const string name="Button", // nome do botão
                      const string text="Text")   // texto
  {
//--- redefine o valor de erro
```

```
   ResetLastError();
//--- alterar texto do objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": falha ao alterar texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o botão                                                  |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // ID do gráfico
                  const string name="Button") // nome do botão
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o botão
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir o botão! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
      return;
     }
//--- definir a etapa de alterar o tamanho do botão
```

© 2000-2024, MetaQuotes Ltd.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- definir as coordenadas botão e seu tamanho
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- criar o botão
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSi
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- redesenhar o gráfico
   ChartRedraw();
//--- reduzir o botão no loop
   int i=0;
   while(i<13)
     {
      //--- meio segundo de atraso
      Sleep(500);
      //--- mudar o botão para o estado pressionado
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- redesenhar o gráfico e esperar por 0.2 segundos
      ChartRedraw();
      Sleep(200);
      //--- redefinir as coordenadas e tamanho do botão
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- reduzir o botão
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- trazer o botão de volta ao estado liberado
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- redesenhar o gráfico
      ChartRedraw();
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- incrementar um contador de loop
      i++;
     }
//--- meio segundo de atraso
   Sleep(500);
//--- excluir o botão
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- esperar por um segundo
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_CHART

Objeto Chart.

Observação

Objetos do tipo "OBJ_CHART" não são suportados (não são exibidos) durante o teste visual.

Coordenadas do ponto de ancoragem são em pixels. Você também pode selecionar uma ancoragem do bitmap no canto a partir do enumerador ENUM_BASE_CORNER.

Símbolo, período e escala podem ser selecionados para o objeto Chart. Escala de preço e modo de exibição de data também pode ser habilitado/desabilitado.

Exemplo

O seguinte script cria e move o objeto Chart no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria objeto \"Chart\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="Chart";              // Nome do objeto
input string            InpSymbol="EURUSD";           // Símbolo
input ENUM_TIMEFRAMES   InpPeriod=PERIOD_H1;          // Período
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;  // Canto de ancoragem
input int               InpScale=2;                   // Escala
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpDateScale=true;           // Exibição da escala de tempo
input bool              InpPriceScale=true;          // Exibição da escala de preço
input color             InpColor=clrRed;             // Cor da borda quando destacada
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // Estilo da linha quando destacad
input int               InpPointWidth=1;             // Tamanho do ponto para movimento
input bool              InpBack=false;               // Objeto de Fundo
input bool              InpSelection=true;           // Destaque para mover
input bool              InpHidden=true;              // Ocultar na lista de objeto
input long              InpZOrder=0;                 // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criando objeto Chart                                             |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,                // ID do grá
                       const string            name="Chart",              // nome do ob
                       const int               sub_window=0,              // índice da
                       const string            symbol="EURUSD",           // símbolo
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,          // período
                       const int               x=0,                       // coordenada
                       const int               y=0,                       // coordenada
                       const int               width=300,                 // largura
                       const int               height=200,                // altura
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER,  // canto de a
                       const int               scale=2,                   // escala
                       const bool              date_scale=true,           // exibição e
                       const bool              price_scale=true,          // exibição e
                       const color             clr=clrRed,                // a cor da b
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,         // estilo da
                       const int               point_width=1,             // mover tama
                       const bool              back=false,                // no fundo
                       const bool              selection=false,           // destaque p
                       const bool              hidden=true,               // ocultar na
                       const long              z_order=0)                 // prioridade
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar objeto Chart
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar objeto \"Chart\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir coordenadas do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- definir tamanho do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- determinar o canto do gráfico onde as coordenadas do ponto são definidas
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- definir o símbolo
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- definir o período
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- definir a escala
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- exibir (true) ou ocultar (false) a escala de tempo
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- exibir (true) ou ocultar (false) a escala de preço
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- definir a cor da borda quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho do ponto de ancoragem para movimentar o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir o símbolo e período de tempo do objeto Chart             |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,       // ID do gr
                                   const string          name="Chart",     // nome do
                                   const string          symbol="EURUSD",  // símbolo
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // período
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir símbolo e período de tempo do objeto Chart
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": falha para definir o símbolo do objeto \"Chart\"! Código de erro = ",Ge
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
```

```
            ": falha para definir o símbolo do objeto \"Chart\"! Código de erro = ",Ge
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover objeto Chart                                               |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // ID do gráfico (não objeto de um Gra
                     const string name="Chart", // nome do objeto
                     const int    x=0,          // coordenada X
                     const int    y=0)          // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover o objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha ao mover coordenada X do objeto \"Chart\"! Código de erro = ",Get
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha ao mover coordenada Y do objeto \"Chart\"! Código de erro = ",Get
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tamanho do objeto Chart                                |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // ID do gráfico (não objeto de
                           const string name="Chart", // nome do objeto
                           const int    width=300,    // largura
                           const int    height=200)   // altura
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar o tamanho do objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura do objeto \"Chart\" ! Código de erro = ",Get
      return(false);
     }
```

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura do objeto \"Chart\" ! Código de erro = ",GetI
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Retorno do ID do objeto Chart                                    |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // ID do gráfico (não objeto de um Gr
                      const string name="Chart") // nome do objeto
  {
//--- preparar a variável para obter o ID do objeto Chart
   long id=-1;
//--- redefine o valor de erro
   ResetLastError();
//--- obter ID
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": falha ao obter o ID do objeto \"Chart\"! Código de erro = ",GetLastErro
     }
//--- retorno do resultado
   return(id);
  }
//+------------------------------------------------------------------+
//| Deletar o objeto Chart                                           |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // ID do gráfico (não objeto de um G
                       const string name="Chart") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o botão
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao deletar objeto \"Chart\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- obter o número de símbolos na Observação do Mercado
   int  symbols=SymbolsTotal(true);
//--- verificar se o símbolo com um nome especifico está presente na lista de símbolo
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Erro! ",InpSymbol," símbolo não está presente na \"Observação do Mercado\
      return;
     }
//--- verificar a validade dos parâmetros de entrada
   if(InpScale<0 || InpScale>5)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }

//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
      return;
     }
//--- definir as coordenadas do objeto gráfico e seu tamanho
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- criar objeto Chart
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

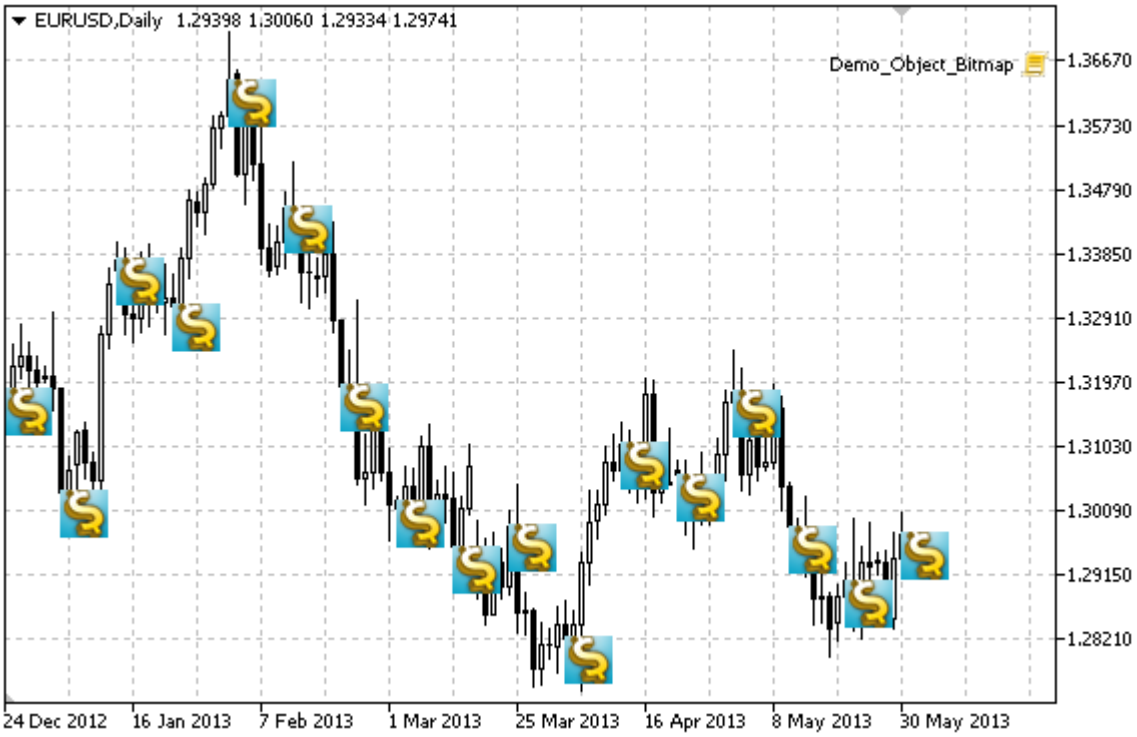
© 2000-2024, MetaQuotes Ltd.

```
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- estender objeto Chart
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- redimensionar
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico e esperar por 0.01 segundos
      ChartRedraw();
      Sleep(10);
     }
//--- meio segundo de atraso
   Sleep(500);
//--- alterar período de tempo do gráfico
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- três segundos de atraso
   Sleep(3000);
//--- excluir o objeto
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- esperar por um segundo
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BITMAP

Objeto Bitmap.



Observação

Para objeto Bitmap, você pode selecionar o escopo de visibilidade de uma imagem.

Exemplo

O seguinte script cria vários bitmaps no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria um bitmap na janela do gráfico."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpFile="\\Images\\dollar.bmp"; // Nome do arquivo Bitmap
input int             InpWidth=24;                    // Coordenada X do escopo de vis
input int             InpHeight=24;                   // Coordenada Y do escopo de vis
input int             InpXOffset=4;                   // Deslocamento do escopo de vis
input int             InpYOffset=4;                   // Deslocamento do escopo de vis
input color           InpColor=clrRed;                // Cor da borda quando destacada
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Estilo de linha quando destac
input int             InpPointWidth=1;                // Tamanho do ponto para movimen
input bool            InpBack=false;                  // Objeto de Fundo
input bool            InpSelection=false;             // Destaque para mover
```

© 2000-2024, MetaQuotes Ltd.

```
input bool            InpHidden=true;                 // Ocultar na lista de objeto
input long            InpZOrder=0;                    // Prioridade para clicar no mou
//+------------------------------------------------------------------+
//| Criar um bitmap na janela do gráfico                             |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // ID do gráfico
                  const string          name="Bitmap",     // nome do bitmap
                  const int             sub_window=0,      // índice da sub-janela
                  datetime              time=0,            // ponto de ancoragem do te
                  double                price=0,           // ponto de ancoragem do pr
                  const string          file="",           // nome do arquivo bitmap
                  const int             width=10,          // coordenada X do escopo d
                  const int             height=10,         // coordenada Y do escopo d
                  const int             x_offset=0,        // escopo de visibilidade d
                  const int             y_offset=0,        // escopo de visibilidade d
                  const color           clr=clrRed,        // cor da borda quando dest
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // estilo de linha quando d
                  const int             point_width=1,     // mover tamanho do ponto
                  const bool            back=false,        // no fundo
                  const bool            selection=false,   // destaque para mover
                  const bool            hidden=true,       // ocultar na lista de obje
                  const long            z_order=0)         // prioridade para clicar n
  {
//--- definir as coordenadas de pontos de ancoragem, se eles não estão definidos
   ChangeBitmapEmptyPoint(time,price);
//--- redefine o valor de erro
   ResetLastError();
//--- criar um bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao criar um bitmap na janela do gráfico! Código de erro = ",GetLa
      return(false);
     }
//--- definir o caminho para o arquivo da imagem
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": falha ao carregar a imagem! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir escopo de visibilidade para a imagem, se os valores da largura ou altura
//--- excederem a largura e a altura (respectivamente) de uma imagem de origem,
//--- não será desenhada, no caso oposto
//--- apenas a parte correspondente a esses valores será desenhada
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- definir a parte de uma imagem que está sendo exibida no escopo de visibilidade,
//--- a parte padrão é a área superior esquerda de uma imagem, os valores permitem
```

© 2000-2024, MetaQuotes Ltd.

```
//--- realizar uma mudança a partir desta área de exibição de uma outra parte da image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- definir a cor da borda quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho do ponto de ancoragem para movimentar o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir uma nova imagem para o bitmap                            |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // ID do gráfico
                    const string name="Bitmap", // nome do bitmap
                    const string file="")       // caminho para o arquivo
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir o caminho para o arquivo da imagem
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": falha ao carregar a imagem! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Criar um bitmap na janela do gráfico                             |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // ID do gráfico
                const string name="Bitmap", // nome do bitmap
                datetime     time=0,        // ponto de ancoragem do tempo
                double       price=0)       // ponto de ancoragem do preço
  {
//--- se a posição do ponto não está definida, mover para a barra atual tendo o preço
```

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- redefine o valor de erro
   ResetLastError();
//--- mover o ponto de ancoragem
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": falha ao mover o ponto de ancoragem! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tamanho do escopo de visibilidade (objeto)               |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,    // ID do gráfico
                      const string name="Bitmap", // nome do bitmap
                      const int    width=0,       // largura do bitmap
                      const int    height=0)      // altura do bitmap
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tamanho do bitmap
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura do bitmap! Código de erro = ",GetLastError()
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura do bitmap! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mudar coordenada canto superior esquerdo do escopo de visibilidade |
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,    // ID do gráfico
                           const string name="Bitmap", // nome do bitmap
                           const int    x_offset=0,    // coordenada X do escopo de v
                           const int    y_offset=0)    // coordenada Y do escopo de v
```

```
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar as coordenadas do escopo de visibilidade do bitmap
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": falha para alterar coordenada X do escopo de visibilidade! Código de er
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": falha para alterar coordenada Y do escopo de visibilidade! Código de er
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir um bitmap                                                |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,    // ID do gráfico
                  const string name="Bitmap") // nome do bitmap
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir um bitmap! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar valores de ponto de ancoragem e definir valores padrão |
//| para aqueles vazios                                              |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- se o tempo do ponto não está definido, será na barra atual
   if(!time)
      time=TimeCurrent();
//--- se o preço do ponto não está definido, ele terá valor Bid
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // array para armazenar datas das barras visíveis
   double   close[]; // array para armazenar Fechamento de preços
//--- nome do arquivo bitmap
   string   file="\\Images\\dollar.bmp";
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- alocação de memória
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- preencher o array dos preços Baixos
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Falha ao copiar os valores dos preços Baixos! Código de erro = ",GetLastE
      return;
     }
//--- definir como frequentemente as imagens devem ser exibidas
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- definir o passo
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- criar bitmaps para valores de barra Alta e Baixa (com intervalos)
   for(int i=0;i<bars;i+=step)
     {
      //--- criar os bitmaps
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- meio segundo de atraso
   Sleep(500);
//--- excluir sinal Sell
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

Objeto Etiqueta Bitmap.

Observação

Posição do ponto de ancoragem relativa a etiqueta pode ser selecionada do enumerador ENUM_ANCHOR_POINT. Coordenadas do ponto de ancoragem são em pixels.

Você também pode selecionar uma ancoragem do bitmap no canto a partir da enumeração ENUM_BASE_CORNER.

Para etiqueta bitmap, você pode selecionar no escopo de visibilidade de uma imagem.

Exemplo

O seguinte script cria vários bitmaps no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script criando objeto \"Bitmap Label\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string            InpName="BmpLabel";                // Nome etiqueta
input string            InpFileOn="\\Images\\dollar.bmp"; // O nome do arquivo para o
input string            InpFileOff="\\Images\\euro.bmp";  // O nome do arquivo para o
input bool              InpState=false;                    // Etiqueta pressionada/libe
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;       // Canto do gráfico para anc
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;           // Tipo de ancoragem
```

© 2000-2024, MetaQuotes Ltd.

```
input color             InpColor=clrRed;                  // A cor da borda quando des
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // Estilo da linha quando de
input int               InpPointWidth=1;                  // Tamanho do ponto para mov
input bool              InpBack=false;                    // Objeto de fundo
input bool              InpSelection=false;               // Destaque para mover
input bool              InpHidden=true;                   // Ocultar na lista de objet
input long              InpZOrder=0;                      // Prioridade para clicar no
//+------------------------------------------------------------------+
//| Criar objeto Etiqueta Bitmap                                     |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // ID do grá
                       const string            name="BmpLabel",          // nome da et
                       const int               sub_window=0,             // índice da
                       const int               x=0,                      // coordenada
                       const int               y=0,                      // coordenada
                       const string            file_on="",               // imagem em
                       const string            file_off="",              // imagem em
                       const int               width=0,                  // coordenada
                       const int               height=0,                 // coordenada
                       const int               x_offset=10,              // escopo de
                       const int               y_offset=10,              // escopo de
                       const bool              state=false,              // pressionad
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // canto do g
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // tipo de an
                       const color             clr=clrRed,               // a cor da b
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // estilo da
                       const int               point_width=1,            // mover tama
                       const bool              back=false,               // no fundo
                       const bool              selection=false,          // destaque p
                       const bool              hidden=true,              // ocultar na
                       const long              z_order=0)                // prioridade
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma etiqueta bitmap
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar objeto \"Etiqueta Bitmap\"! Código de erro = ",GetLastEr
      return(false);
     }
//--- definir as imagens para os modos ligar (On) e desligar (Off)
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": falha para carregar a imagem para o modo On! Código de erro = ",GetLast
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
```

```
      {
       Print(__FUNCTION__,
             ": falha para carregar a imagem para o modo Off! Código de erro = ",GetLas
       return(false);
      }
//--- definir coordenadas da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- definir escopo de visibilidade para a imagem, se os valores da largura ou altura
//--- excederem a largura e a altura (respectivamente) de uma imagem de origem,
//--- não será desenhada, no caso oposto
//--- apenas a parte correspondente a esses valores será desenhada
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- definir a parte de uma imagem que está sendo exibida no escopo de visibilidade,
//--- a parte padrão é a área superior esquerda de uma imagem, os valores permitem
//--- realizar uma mudança a partir desta área de exibição de uma outra parte da image
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- definir os status da etiqueta(pressionada ou liberada)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- determinar o canto do gráfico onde as coordenadas do ponto são definidas
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- tipo de definição de ancoragem
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- definir a cor da borda quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo de linha quando o modo de destaque do objeto é habilitado
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir o tamanho do ponto de ancoragem para movimentar o objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Definir uma nova imagem para objeto etiqueta Bitmap              |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,       // ID do gráfico
                         const string name="BmpLabel", // nome da etiqueta
                         const int    on_off=0,        // modificador (On ou Off)
```

```
                          const string file="")         // caminho para o arquivo
  {
//--- redefine o valor de erro
   ResetLastError();
//--- definir o caminho para o arquivo da imagem
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": falha para carregar a imagem! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover objeto Etiqueta Bitmap                                     |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // ID do gráfico
                     const string name="BmpLabel", // nome da etiqueta
                     const int    x=0,             // coordenada X
                     const int    y=0)             // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover o objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada Y do objeto! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar tamanho do escopo de visibilidade (objeto)               |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // ID do gráfico
                           const string name="BmpLabel", // nome da etiqueta
                           const int    width=0,         // largura da etiqueta
                           const int    height=0)        // altura da etiqueta
  {
//--- redefine o valor de erro
```

```
   ResetLastError();
//--- alterar o tamanho do objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura do objeto! Código de erro = ",GetLastError()
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura do objeto! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mudar coordenada canto superior esquerdo do escopo de visibilidade |
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID do grafico
                                const string name="BmpLabel", // nome da etiqueta
                                const int    x_offset=0,      // coordenada X do escop
                                const int    y_offset=0)      // coordenada Y do escop
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar as coordenadas do escopo de visibilidade do objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": falha para alterar coordenada X do escopo de visibilidade! Código de er
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": falha para alterar coordenada Y do escopo de visibilidade! Código de er
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir o Objeto "Bitmap label"                                  |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID do gráfico
                       const string name="BmpLabel") // nome da etiqueta
  {
```

```
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao excluir objeto \"Bitmap label\"! Código de erro = ",GetLastEr
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
      return;
     }
//--- definir coordenadas da etiqueta bitmap
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- definir o tamanho da etiqueta e a coordenada do escopo de visibilidade
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- local da etiqueta bitmap no centro da janela
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- redesenhar o gráfico e esperar um segundo
   ChartRedraw();
   Sleep(1000);
//--- alterar tamanho do escopo de visibilidade da etiqueta em loop
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0;i<6;i++)
     {
      //--- alterar o tamanho do escopo
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.3 segundos de atraso
      Sleep(300);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar as coordenadas do escopo de visibilidade da etiqueta em loop
   for(int i=0;i<2;i++)
     {
      //--- alterar coordenadas do escopo de visibilidade
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.3 segundos de atraso
      Sleep(300);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir a etiqueta
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EDIT

Objeto Edit.

Observação

Coordenadas do ponto de ancoragem são em pixels. Você também pode selecionar uma ancoragem do Edit no canto a partir do enumerador ENUM_BASE_CORNER.

Você também pode selecionar um dos tipos de alinhamento de texto dentro de Edit a partir da enumeração ENUM_ALIGN_MODE.

Exemplo

O seguinte script cria e move o objeto Edit no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria objeto \"Edit\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string           InpName="Edit";              // Nome do objeto
input string           InpText="Text";              // Texto de objeto
input string           InpFont="Arial";             // Fonte
input int              InpFontSize=14;              // Tamanho da fonte
input ENUM_ALIGN_MODE  InpAlign=ALIGN_CENTER;       // Tipo de alinhamento de texto
input bool             InpReadOnly=false;           // Habilidade de editar
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Canto do gráfico para ancoragem
input color            InpColor=clrBlack;           // Cor do texto
```

© 2000-2024, MetaQuotes Ltd.

```
input color            InpBackColor=clrWhite;       // Fundo da cor
input color            InpBorderColor=clrBlack;     // Cor da borda
input bool             InpBack=false;               // Objeto de Fundo
input bool             InpSelection=false;          // Destaque para mover
input bool             InpHidden=true;              // Ocultar na lista de objeto
input long             InpZOrder=0;                 // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar o objeto Edit                                              |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,                // ID do gráfico
                const string           name="Edit",               // nome do objeto
                const int              sub_window=0,              // índice da sub-jane
                const int              x=0,                       // coordenada X
                const int              y=0,                       // coordenada Y
                const int              width=50,                  // largura
                const int              height=18,                 // altura
                const string           text="Text",               // texto
                const string           font="Arial",              // fonte
                const int              font_size=10,              // tamanho da fonte
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,        // tipo de alinhament
                const bool             read_only=false,           // habilidade para ed
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER,  // canto do gráfico p
                const color            clr=clrBlack,              // cor do texto
                const color            back_clr=clrWhite,         // cor do fundo
                const color            border_clr=clrNONE,        // cor da borda
                const bool             back=false,                // no fundo
                const bool             selection=false,           // destaque para move
                const bool             hidden=true,               // ocultar na lista d
                const long             z_order=0)                 // prioridade para cl
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar campo de edição
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar objeto \"Edit\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir coordenadas do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- definir tamanho do objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- definir o texto
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- definir o texto fonte
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir tamanho da fonte
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- definir o tipo de alinhamento do texto no objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- habilitar (true) ou cancelar (false) modo de somente leitura
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- definir o canto do gráfico onde as coordenadas do objeto são definidas
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- definir a cor do texto
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a cor de fundo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- definir a cor da borda
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover objeto Edit                                                |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,   // ID do gráfico
              const string name="Edit", // nome do objeto
              const int    x=0,         // coordenada X
              const int    y=0)         // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover o objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada Y do objeto! Código de erro = ",GetLastErro
      return(false);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Redimensionar objeto Edit                                        |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // ID do gráfico
                    const string name="Edit", // nome do objeto
                    const int    width=0,     // largura
                    const int    height=0)    // altura
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar o tamanho do objeto
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura do objeto! Código de erro = ",GetLastError()
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura do objeto! Código de erro = ",GetLastError())
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar texto do objeto Edit                                     |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // ID do gráfico
                    const string name="Edit", // nome do objeto
                    const string text="Text") // texto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar texto do objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": falha ao alterar texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Retornar texto de objeto                                          |
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,        // texto
                 const long   chart_ID=0,  // ID do gráfico
                 const string name="Edit") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- obter texto do objeto
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": falha ao obter o texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir objeto Edit                                               |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // ID do gráfico
                const string name="Edit") // nome do objeto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": falha ao deletar objeto \"Edit\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
       return;
      }
//--- definir o passo para mudar o campo edit
   int x_step=(int)x_distance/64;
//--- definir coordenadas do campo edit e seu tamanho
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- armazenar o texto na variável local
   string text=InpText;
//--- criar campo de edição
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,I
      InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- estender o campo edit
   while(x_size-x<x_distance*5/8)
      {
       //--- aumentar a largura do campo edit
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- verificar se o funcionamento do script foi desativado a força
       if(IsStopped())
          return;
       //--- redesenhar o gráfico e esperar por 0.05 segundos
       ChartRedraw();
       Sleep(50);
      }
//--- meio segundo de atraso
   Sleep(500);
//--- alterar o texto
   for(int i=0;i<20;i++)
      {
       //--- adicionar "+" no início e no final
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- verificar se o funcionamento do script foi desativado a força
       if(IsStopped())
```

© 2000-2024, MetaQuotes Ltd.

```
          return;
      //--- redesenhar o gráfico e esperar por 0.1 segundos
      ChartRedraw();
      Sleep(100);
     }
//--- meio segundo de atraso
   Sleep(500);
//--- excluir campo edit
   EditDelete(0,InpName);
   ChartRedraw();
//--- esperar por um segundo
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_EVENT

Objeto Evento.

Observação

Quando mover o mouse sobre o evento, o texto aparece.

Exemplo

O seguinte script cria e move o objeto Evento no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script desenha objeto gráfico \"Evento\"."
#property description "Coordenadas do ponto de ancoragem é configurado em porcentagem
#property description "largura da janela do gráfico em barras."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string          InpName="Event";     // Nome do evento
input int             InpDate=25;          // Data do Evento, %
input string          InpText="Text";      // Texto do Evento
input color           InpColor=clrRed;     // Cor do Evento
input int             InpWidth=1;          // Tamanho do ponto quando destacado
input bool            InpBack=false;       // Evento de Fundo
input bool            InpSelection=false;  // Destaque para mover
input bool            InpHidden=true;      // Ocultar na lista de objeto
```

© 2000-2024, MetaQuotes Ltd.

```
input long            InpZOrder=0;        // Prioridade para clicar no mouse
//+------------------------------------------------------------------+
//| Create objeto Evento no gráfico                                  |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // ID de gráfico
                 const string          name="Event",     // nome do evento
                 const int             sub_window=0,     // índice da sub-janela
                 const string          text="Text",      // texto do evento
                 datetime              time=0,           // tempo
                 const color           clr=clrRed,       // cor
                 const int             width=1,          // largura do ponto quando em
                 const bool            back=false,       // no fundo
                 const bool            selection=false,  // destaque para mover
                 const bool            hidden=true,      // ocultar na lista de objetos
                 const long            z_order=0)        // prioridade para clique do r
  {
//--- se o tempo não estiver definido, criar o objeto no último bar
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- criar objeto Evento
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar objeto \"Evento\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- definir texto do evento
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- definir cor
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir a largura de ponto de ancoragem se o objeto é destacado
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- permitir (true) ou desabilitar (false) o modo de movimento do evento com o mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o texto do objeto Evento                                 |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // ID do gráfico
                     const string name="Event", // nome do evento
                     const string text="Text")  // texto
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar texto do objeto
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": falha ao alterar texto! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover objeto Evento                                              |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // ID do gráfico
               const string name="Event", // nome do evento
               datetime     time=0)       // tempo
  {
//--- se o tempo não estiver definido, mover evento até o último bar
   if(!time)
      time=TimeCurrent();
//--- redefine o valor de erro
   ResetLastError();
//--- mover o objeto
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": falha ao mover objeto \"Evento\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir objeto Evento                                            |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // ID do gráfico
                 const string name="Event") // nome do evento
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir o objeto
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": falha ao deletar objeto \"Evento\"! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- verificar a exatidão dos parâmetros de entrada
   if(InpDate<0 || InpDate>100)
     {
      Print("Erro! Valores incorretos dos parâmetros de entrada!");
      return;
     }
//--- número de barras visíveis na janela do gráfico
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- array para armazenar a data de valores a serem utilizados
//--- para definir e alterar as coordenadas de pontos de ancoragem
   datetime date[];
//--- alocação de memória
   ArrayResize(date,bars);
//--- preencher o array das datas
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Falha ao copiar valores de tempo! Código de erro = ",GetLastError());
      return;
     }
//--- definir os pontos para criar um objeto
   int d=InpDate*(bars-1)/100;
//--- criar objeto Evento
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- agora, mover o objeto
//--- contador de loop
   int h_steps=bars/2;
//--- mover o objeto
   for(int i=0;i<h_steps;i++)
     {
```

```
      //--- usar o seguinte valor
      if(d<bars-1)
         d+=1;
      //--- mover o ponto
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico
      ChartRedraw();
      // 0.05 segundos de atraso
      Sleep(50);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- excluir o canal do gráfico
   EventDelete(0,InpName);
   ChartRedraw();
//--- 1 segundo de atraso
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

Objeto Etiqueta Retangular.

Observação

Coordenadas do ponto de ancoragem são em pixels. É possível a ancoragem no canto de uma etiqueta retangular a partir do enumerador ENUM_BASE_CORNER. O tipo de borda da etiqueta retangular pode ser selecionado a partir do enumerador ENUM_BORDER_TYPE.

O objeto é usado para criar e projetar a interface gráfica personalizada.

Exemplo

O seguinte script cria e move o objeto Etiqueta Retangular no gráfico. Funções especiais têm sido desenvolvidas para criar e alterar as propriedades do objeto gráfico. Você pode utilizar estas funções "as is" em seus próprios aplicativos.

```
//--- descrição
#property description "Script cria objeto gráfico \"Etiqueta Retangular\"."
//--- janela de exibição dos parâmetros de entrada durante inicialização do script
#property script_show_inputs
//--- entrada de parâmetros do script
input string           InpName="RectLabel";          // Nome etiqueta
input color            InpBackColor=clrSkyBlue;      // Fundo da cor
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Tipo de Borda
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Canto do gráfico para ancoragem
input color            InpColor=clrDarkBlue;         // Cor da borda plana (Flat)
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // Estilo da borda plana (Flat)
input int              InpLineWidth=3;               // Largura da borda plana (Flat)
```

© 2000-2024, MetaQuotes Ltd.

```
input bool              InpBack=false;                // Objeto de Fundo
input bool              InpSelection=true;            // Destaque para mover
input bool              InpHidden=true;               // Ocultar na lista de objeto
input long              InpZOrder=0;                  // Prioridade para clique do mouse
//+------------------------------------------------------------------+
//| Criar etiqueta retangular                                        |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,                // ID do gráfico
                     const string           name="RectLabel",          // nome da etiqu
                     const int              sub_window=0,              // índice da sub
                     const int              x=0,                       // coordenada X
                     const int              y=0,                       // coordenada Y
                     const int              width=50,                  // largura
                     const int              height=18,                 // altura
                     const color            back_clr=C'236,233,216',   // cor do fundo
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,      // tipo de borda
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER,  // canto do grá
                     const color            clr=clrRed,                // cor da borda
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,         // estilo da bor
                     const int              line_width=1,              // largura da bo
                     const bool             back=false,                // no fundo
                     const bool             selection=false,           // destaque para
                     const bool             hidden=true,               // ocultar na li
                     const long             z_order=0)                 // prioridade pa
  {
//--- redefine o valor de erro
   ResetLastError();
//--- criar uma etiqueta retangular
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": falha ao criar uma etiqueta retangular! Código de erro = ",GetLastError
      return(false);
     }
//--- definir coordenadas da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- definir tamanho da etiqueta
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- definir a cor de fundo
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- definir o tipo de borda
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- determinar o canto do gráfico onde as coordenadas do ponto são definidas
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- definir a cor da borda plana (no modo Flat)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- definir o estilo da linha da borda plana
```

© 2000-2024, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- definir a largura da borda plana
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- exibir em primeiro plano (false) ou fundo (true)
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Habilitar (true) ou desabilitar (false) o modo de movimento da etiqueta pelo mou
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- ocultar (true) ou exibir (false) o nome do objeto gráfico na lista de objeto
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- definir a prioridade para receber o evento com um clique do mouse no gráfico
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Mover a etiqueta retangular                                      |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // ID do gráfico
                   const string name="RectLabel", // nome da etiqueta
                   const int    x=0,              // coordenada X
                   const int    y=0)              // coordenada Y
  {
//--- redefine o valor de erro
   ResetLastError();
//--- mover a etiqueta retangular
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": falha para mover coordenada X do objeto! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tamanho da etiqueta retangular                         |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // ID do gráfico
                         const string name="RectLabel", // nome da etiqueta
                         const int    width=50,         // largura da etiqueta
                         const int    height=18)        // altura da etiqueta
  {
```

```
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tamanho da etiqueta
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a largura da etiqueta! Código de erro = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": falha ao alterar a altura da etiqueta! Código de erro = ",GetLastError
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Alterar o tipo de borda da etiqueta retangular                   |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // ID do g
                               const string           name="RectLabel",     // nome da
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // tipo de
  {
//--- redefine o valor de erro
   ResetLastError();
//--- alterar tipo de borda
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": falha ao alterar tipo de borda! Código de erro = ",GetLastError());
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluir etiqueta retangular                                      |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // ID do gráfico
                     const string name="RectLabel") // nome da etiqueta
  {
//--- redefine o valor de erro
   ResetLastError();
//--- excluir a etiqueta
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2024, MetaQuotes Ltd.

```
            ": falha ao excluir a etiqueta retangular! Código de erro = ",GetLastErro
      return(false);
     }
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tamanho da janela do gráfico
   long x_distance;
   long y_distance;
//--- definir tamanho da janela
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Falha ao obter a largura do gráfico! Código de erro = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Falha ao obter a altura do gráfico! Código de erro = ",GetLastError());
      return;
     }
//--- definir as coordenadas da etiqueta retangular
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- definir tamanho da etiqueta
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- criar uma etiqueta retangular
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- redesenhar o gráfico e esperar um segundo
   ChartRedraw();
   Sleep(1000);
//--- alterar o tamanho da etiqueta retangular
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- redimensionar
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

```
      //--- verificar se o funcionamento do script foi desativado a força
      if(IsStopped())
         return;
      //--- redesenhar o gráfico e esperar por 0.01 segundos
      ChartRedraw();
      Sleep(10);
     }
//--- 1 segundo de atraso
   Sleep(1000);
//--- alterar tipo de borda
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- alterar tipo de borda
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- redesenhar o gráfico e esperar por um segundo
   ChartRedraw();
   Sleep(1000);
//--- excluir a etiqueta
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- esperar por um segundo
   Sleep(1000);
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

Propriedades do Objeto

Os objetos gráficos podem ter muitas propriedades dependendo do tipo de objeto. A configuração e a obtenção de valores das propriedades dos objetos são efetuadas de acordo com as funções de trabalho com objetos gráficos.

Todos os objetos usados na análise técnica são vinculados aos gráficos nas coordenadas de preço e tempo: linha de tendência, canais, ferramentas de Fibonacci, etc. Mas há uma série de objetos de apoio projetados para melhorar a interface, e que estão vinculados à parte sempre visível do gráfico (janela principal ou subjanela do gráfico do indicador):

Objeto	Identificador	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Sim	—	Sim	Sim
Label	OBJ_LABEL	Sim	Sim (apenas leitura)	—	Sim	Sim	Sim
Button	OBJ_BUTTON	Sim	Sim	—	Sim	—	—
Bitmap	OBJ_BITMAP	—	Sim (apenas leitura)	Sim	—	Sim	—
Bitmap Label	OBJ_BITMAP_LABEL	Sim	Sim (apenas leitura)	—	Sim	Sim	—
Edit	OBJ_EDIT	Sim	Sim	—	Sim	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Sim	Sim	—	Sim	—	—

Na tabela são utilizadas as seguintes denominações:

- **X/Y** - as coordenadas do ponto âncora são especificadas em pixels em relação a um dos cantos do gráfico;
- **Width/Height** - os objetos têm largura e altura. Se você definir "apenas leitura", isso significa que os valores para largura e altura são calculados somente após o processamento do objeto no gráfico;
- **Date/Price** - as coordenadas do ponto âncora são definidas pelo par data/preço;
- **OBJPROP_CORNER** - define o ângulo do gráfico, que especifica as coordenadas do ponto de âncora. Pode ser um dos 4 valores de enumeração ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - define a posição do ponto de âncora no objeto, e pode ser um dos 9 valores de enumeração ENUM_ANCHOR_POINT. Precisamente a partir desse ponto até ao canto selecionado são expecificadas as cordenadas em pixels;

© 2000-2024, MetaQuotes Ltd.

- **OBJPROP_ANGLE** – define o ângulo de rotação do objeto no sentido anti-horário.

As funções que definem as propriedades dos objetos gráficos, bem como as operações ObjectCreate() e ObjectMove() para criar e mover objetos ao longo do gráfico são, na verdade, usadas para enviar comandos para o gráfico. Se essas funções forem executadas com êxito, o comando será incluído na fila comum dos eventos do gráfico. As mudanças visuais nas propriedades dos objetos gráficos são implementadas ao lidar com a fila dos eventos do gráfico.

Assim, não espere uma atualização visual imediata de objetos gráficos após chamar essas funções. Geralmente, os objetos gráficos no gráfico são atualizados automaticamente pelo terminal após a mudança de eventos - a chegada de uma nova cotação, o redimensionamento da janela de gráfico, etc. Use a função ChartRedraw() para forçar a atualização dos objetos gráficos.

Para funções ChartSetDouble() e ChartGetDouble()

ENUM_OBJECT_PROPERTY_INTEGER

Identificador	Descrição	Tipo de Propriedade
OBJPROP_COLOR	Cor	color
OBJPROP_STYLE	Estilo	ENUM_LINE_STYLE
OBJPROP_WIDTH	Espessura de linha	int
OBJPROP_BACK	Objeto no pano de fundo	bool
OBJPROP_ZORDER	Prioridade de um objeto gráfico para receber eve	long

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	ntos de clicagem em um gráfico (CHART_EVENT_CLICK). O valor zero default é definido ao se criar um objeto; a prioridade pode ser aumentada se necessário. Ao colocar obje	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	tos um sobre o outro, somente aquele com a maior prioridade receberá o evento CHARTEVENT_CLICK.	
OBJPROP_FILL	Preenche um objeto com cor (para OBJ_RECTANGLE, OBJ_TRIANGLE	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_HIDDEN	Proíbe exibir o nome de um objeto gráfico na lista de objetos do menu do terminal "Gráficos" - "Objetos	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	" - "Lista de objetos". O valor true permite ocultar um objeto da lista. Por default, true é definido para os objetos que exibem eventos de calendário, histórico de negociação	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	e para objetos criados a partir de programas MQL5. Para ver tais objetos gráficos e acessar suas propriedades, clique no botão "TODOS" na janela "Lista de objetos".	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
OBJPROP_SELECTED	Objeto é selecionado	bool
OBJPROP_READONLY	Capacidade de editar texto no objeto Edit	bool
OBJPROP_TYPE	Tipo de objeto	ENUM_OBJECT r/o
OBJPROP_TIME	Coordenada de tempo	datetime modificador=número do ponto de ancoragem
OBJPROP_SELECTABLE	Disponibilidade de objeto	bool
OBJPROP_CREATETIME	Hora de criação do objeto	datetime r/o
OBJPROP_LEVELS	Número de	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	níveis	
OBJPROP_LEVELCOLOR	Cor das linhas de nível	color modificador=número de nível
OBJPROP_LEVELSTYLE	Estilo da linha de nível	ENUM_LINE_STYLE modificador=número de nível
OBJPROP_LEVELWIDTH	Espessura do nível de linha	int modificador=número de nível
OBJPROP_ALIGN	Alinhamento de texto horizontal no objeto "Edit" (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	Tamanho de fonte	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
OBJPROP_RAY_LEFT	Raios indo para esquerda	bool
OBJPROP_RAY_RIGHT	Raios indo para direita	bool
OBJPROP_RAY	Uma linha vertical indo através de todas as janelas de um gráfico	bool
OBJPROP_ELLIPSE	Exibição da elipse completa do objeto Arco de Fibonacci	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	(OBJ_FIBOARC)	
OBJPROP_ARROWCODE	Código de seta para o objeto Seta	uchar
OBJPROP_TIMEFRAMES	Visibilidade de um objeto nas janelas de tempo	conjunto de flags
OBJPROP_ANCHOR	Localização do ponto de ancoragem de um objeto gráfico	ENUM_ARROW_ANCHOR (for OBJ_ARROW), ENUM_ANCHOR_POINT (for OBJ_LABEL, OBJ_BITMAP_LABEL and OBJ_TEXT)
OBJPROP_XDISTANCE	A distância	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	em pixels ao longo do eixo X a partir do canto de vinculação (see note)	
OBJPROP_YDISTANCE	A distância em pixels ao longo do eixo Y a partir do canto de vinculação (see note)	int
OBJPROP_DIRECTION	Tendência do	ENUM_GANN_DIRECTION

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	objeto Gann	
OBJPROP_DEGREE	Nível de marcação da Onda de Elliott	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Exibição de linhas para marcação da Onda de Elliott	bool
OBJPROP_STATE	Estado do botão (pressionado / não pressionado)	bool
OBJPROP_CHART_ID	ID do objeto "Chart"	long r/o

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	(OBJ_CHART). Permite trabalhar com as propriedades deste objeto como um gráfico normal usando as funções descritas em Operações de Gráfico, mas existem algumas	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	exceções.	
OBJPROP_XSIZE	O tamanho do objeto Chart ao longo do eixo X (largura em pixels) é especificado para objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP,	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_YSIZE	O tamanho do objeto Chart ao longo do eixo X (largura em pixels) é especificado para objetos OBJ_LABEL (read only),	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_XOFFSET	A coordenada X do canto superior esquerdo da área visível retangular nos	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	objetos gráficos "Bitmap Label" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.	
OBJPROP_YOFFSET	A coorden	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	ada X do canto superior esquerdo da área visível retangular nos objetos gráficos "BitmapLabel" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). O valor é definido em pixels em rela	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	ção ao canto superior esquerdo da imagem original.	
OBJPROP_PERIOD	Janela de tempo para o objeto Chart	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Exibição da escala de tempo para o objeto Chart	bool
OBJPROP_PRICE_SCALE	Exibição da escala de	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	preço para o objeto Chart	
OBJPROP_CHART_SCALE	A escala para o objeto Chart	int valor na faixa de 0-5
OBJPROP_BGCOLOR	A cor de fundo para OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	color
OBJPROP_CORNER	O canto do gráfico para linkar	ENUM_BASE_CORNER

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	um objeto gráfico	
OBJPROP_BORDER_TYPE	Tipo de borda para o objeto "Rectangle label"	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	Cor de borda para os objetos OBJ_EDIT e OBJ_BUTTON	color

Ao usar operações de gráfico para o objeto "Chart" (OBJ_CHART), as seguintes limitações são impostas:

- Ele não pode ser fechado usando ChartClose();
- Ativo/período não podem ser alterados usando a função ChartSetSymbolPeriod();
- As seguintes propriedades são sem efeito: CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE e CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

Para os objetos OBJ_BITMAP_LABEL e OBJ_BITMAP, um modo especial de exibição de imagem pode ser definido por programação. Neste modo, somente parte da imagem original (na qual uma área

© 2000-2024, MetaQuotes Ltd.

visível retangular é aplicada) é exibida, enquanto o resto da imagem fica invisível. O tamanho desta área deve ser definido usando a propriedade OBJPROP_XSIZE e OBJPROP_YSIZE. A área visível pode ser "movida" somente dentro da imagem original usando as propriedades OBJPROP_XOFFSET e OBJPROP_YOFFSET.

Para objetos de tamanho fixo: OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT e OBJ_CHART propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE defina a posição do canto superior esquerdo do ponto do objeto em relação ao canto de gráfico (OBJPROP_CORNER), a partir das quais as coordenadas X e Y serão contadas em pixels.

Para as funções ObjectSetDouble() e ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

Identificador	Descrição	Tipo de Propriedade
OBJPROP_PRICE	Coordenada de Preço	double modificador=número do ponto de ancoragem
OBJPROP_LEVELVALUE	Valor de nível	double modificador=número de nível
OBJPROP_SCALE	Escala (propriedades de objetos Gann e Arcos de Fibonnacci)	double
OBJPROP_ANGLE	Ângulo.	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	Para os objetos sem ângulo especificado, criado a partir de um programa, o valor é igual a EMPTY_VALUE	
OBJPROP_DEVIATION	Desvio para o Canal de Desvio Padrão	double

Para as funções ObjectSetString() e ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
OBJPROP_NAME	Nome de objeto	string
OBJPROP_TEXT	Descrição do objeto (o texto contido no objeto)	string
OBJPROP_TOOLTIP	O texto de um "tooltip" (dica). Se a propriedade não é definida, então o "tooltip" gerado automaticamente	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	pelo terminal é exibido. Um "tooltip" pode ser desabilitado através da atribuição do valor "\n" (quebra de linha) a ele	
OBJPROP_LEVELTEXT	Nível de descrição	string modificador=número de nível
OBJPROP_FONT	Font	string
OBJPROP_BMPFILE	O nome do arquivo BMP para rótu	string modificador: 0-estado Ligado, 1-estado Desligado

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo de Propriedade
	lo Bitmap. Veja também Recursos	
OBJPROP_SYMBOL	Ativo para o objeto Chart	string

Para o objeto OBJ_RECTANGLE_LABEL ("Rótulo de Retângulo") um dentre os três modos de design podem ser definidos, correspondendo aos valores de ENUM_BORDER_TYPE.

ENUM_BORDER_TYPE

Identificador	Descrição
BORDER_FLAT	Forma lisa
BORDER_RAISED	Forma em relevo
BORDER_SUNKEN	Forma cavada

Para o objeto OBJ_EDIT ("Edit") e para a função ChartScreenShot(), pode-se especificar o tipo de alinhamento horizontal usando os valores da enumeração ENUM_ALIGN_MODE.

ENUM_ALIGN_MODE

Identificador	Descrição
ALIGN_LEFT	Alinhamento a esquerda
ALIGN_CENTER	Centrado (somente para o objeto Edit)
ALIGN_RIGHT	Alinhamento a direita

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Objeto ",label_name," não encontrado. Código de erro = ",GetLastError());
      //--- cria objeto Rótulo
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- define coordenada X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- define coordenada Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- define cor de texto
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- define texto para objeto Rótulo
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- define fonte
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- define tamanho de fonte
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- rotação de 45 graus em sentido horário
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- desabilita seleção de mouse
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- desenha o objeto no gráfico
      ChartRedraw(0);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Métodos de Vinculação de Objeto

Os objetos de gráficos Text, Label, Bitmap e Bitmap Label (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP e OBJ_BITMAP_LABEL) podem ter uma de 9 maneiras diferentes de vincular suas coordenadas, as quais foram especificadas pela propriedade OBJPROP_ANCHOR.

Objeto	Identificador	X/Y	Largura/Altura	Data/Preço	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Texto	OBJ_TEXT	—	—	Sim	—	**Sim**	Sim
etiqueta	OBJ_LABEL	Sim	Sim (apenas leitura)	—	Sim	**Sim**	Sim
Botão	OBJ_BUTTON	Sim	Sim	—	Sim	—	—
Bitmap	OBJ_BITMAP	—	Sim (apenas leitura)	Sim	—	**Sim**	—
Bitmap Label	OBJ_BITMAP_LABEL	Sim	Sim (apenas leitura)	—	Sim	**Sim**	—
Edit	OBJ_EDIT	Sim	Sim	—	Sim	—	—
etiqueta retangular	OBJ_RECTANGLE_LABEL	Sim	Sim	—	Sim	—	—

Na tabela são utilizadas as seguintes denominações:

- **X/Y** - as coordenadas do ponto âncora são especificadas em pixels em relação a um dos cantos do gráfico;
- **Width/Height** - os objetos têm largura e altura. Se você definir "apenas leitura", isso significa que os valores para largura e altura são calculados somente após o processamento do objeto no gráfico;
- **Date/Price** - as coordenadas do ponto âncora são definidas pelo par data/preço;
- **OBJPROP_CORNER** - define o ângulo do gráfico, que especifica as coordenadas do ponto de âncora. Pode ser um dos 4 valores de enumeração ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - define a posição do ponto de âncora no objeto, e pode ser um dos 9 valores de enumeração ENUM_ANCHOR_POINT. Precisamente a partir desse ponto até ao canto selecionado são expecificadas as cordenadas em pixels;
- **OBJPROP_ANGLE** - define o ângulo de rotação do objeto no sentido anti-horário.

A variante necessária pode ser especificada usando a função ObjectSetInteger(chart_handle, object_name, OBJPROP_ANCHOR, anchor_point_mode), onde anchor_point_mode é um dos valores de ENUM_ANCHOR_POINT.

ENUM_ANCHOR_POINT

© 2000-2024, MetaQuotes Ltd.

ID	Descrição
ANCHOR_LEFT_UPPER	Ponto de ancoragem no canto superior esquerdo
ANCHOR_LEFT	Ponto de ancoragem para a esquerda no centro
ANCHOR_LEFT_LOWER	Ponto de ancoragem no canto inferior esquerdo
ANCHOR_LOWER	Ponto de ancoragem abaixo no centro
ANCHOR_RIGHT_LOWER	Ponto de ancoragem no canto inferior direito
ANCHOR_RIGHT	Ponto de ancoragem para a direita no centro
ANCHOR_RIGHT_UPPER	Ponto de ancoragem no canto superior direito
ANCHOR_UPPER	Ponto de ancoragem acima no centro
ANCHOR_CENTER	Ponto de ancoragem rigorosamente no centro do objeto

Os objetos OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT e OBJ_CHART possuem um ponto de ancoragem fixo no canto superior esquerdo (ANCHOR_LEFT_UPPER).

Exemplo:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Objeto ",text_name," não encontrado. Código de erro = ",GetLastError());
   //--- Obtém o preço máximo do gráfico
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- Cria o objeto Label (Rótulo)
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- Define cor do texto
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- Define cor de fundo
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- Define texto para o objeto Label (Rótulo)
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- Define fonte de texto
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- Define tamanho de fonte
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- Vincula no canto superior direito
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- Gira 90 graus em sentido anti-horário
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- Proíbe a seleção do objeto pelo mouse
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- redesenha objeto
   ChartRedraw(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Objetos gráficos de Seta (OBJ_ARROW) têm somente 2 formas de linkar suas coordenadas. Os identificadores estão listados em ENUM_ARROW_ANCHOR.

ENUM_ARROW_ANCHOR

ID	Descrição
ANCHOR_TOP	Ancoragem no lado de cima
ANCHOR_BOTTOM	Ancoragem no lado de baixo

Exemplo:

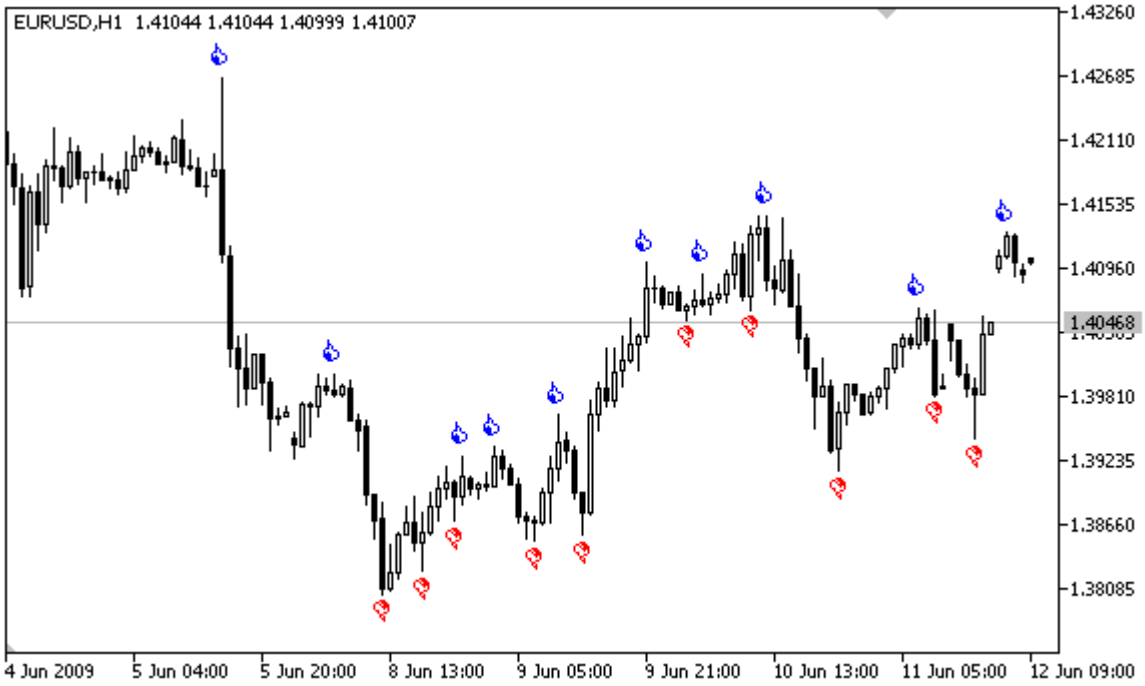
```
void OnStart()
  {
//--- Arrays auxiliares
   double Ups[],Downs[];
   datetime Time[];
//--- Define os arrays como séries de tempo
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- Cria handle do Indicador Fractais
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- Define valor do último erro para Zero
   ResetLastError();
//--- Tenta copiar os valores do indicador
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Não é possível copiar os fractais de alta. Erro = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Tenta copiar os valores do indicador
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Não é possível copiar os fractais de baixa. Erro = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Copia a série de tempo contendo as barras de abertura das últimas 1000 barras
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("Não é possível copiar o tempo de abertura das últimas 1000 barras");
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }

  int upcounter=0,downcounter=0; // conta o número de setas
  bool created;// recebe o resultado das tentativas de criar um objeto
  for(int i=2;i<copied;i++)// Passa pelos valores do indicador iFractals
    {
     if(Ups[i]!=EMPTY_VALUE)// Encontrado o fractal de cima
       {
        if(upcounter<10)// Não cria mais que 10 setas "Up"
          {
           //--- Tenta criar um objeto "Up"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
           if(created)// Se construído - vamos fazer ajuste nele
             {
              //--- Ponto de ancoragem é em baixo, a fim de não cobrir a barra
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
              //--- Toque final - colorido
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
              upcounter++;
             }
          }
       }
     if(Downs[i]!=EMPTY_VALUE)// Encontrado o menor fractal
       {
        if(downcounter<10)// Não cria mais que 10 setas "Down"
          {
           //--- Tenta criar um objeto "Down"
           created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
           if(created)// Se construído - vamos fazer ajuste nele
             {
              //--- Ponto de ancoragem é em acima, a fim de não cobrir a barra
              ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
              //--- Toque final - colorido
              ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
              downcounter++;
             }
          }
       }
    }
}
```

Após a execução do script o gráfico se parecerá com esta figura.

© 2000-2024, MetaQuotes Ltd.



© 2000-2024, MetaQuotes Ltd.

O Canto do Gráfico no Qual um Objeto Fica Ligado

Existe uma série de objetos gráficos, para os quais você pode definir o ângulo, que especifica as coordenadas do ponto de âncora. Esses são os seguintes tipos de objetos (os identificadores de tipo de objeto estão entre parênteses):

- Label (OBJ_LABEL); Rótulo do texto
- Button (OBJ_BUTTON); Botão
- Bitmap Label (OBJ_BITMAP_LABEL); Rótulo do gráfico
- Edit (OBJ_EDIT); Campo de entrada
- Rectangle Label (OBJ_RECTANGLE_LABEL). Rótulo retangular

Objeto	Identificador	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	—	—	Sim	—	Sim	Sim
Label	OBJ_LABEL	Sim	Sim (apenas leitura)	—	**Sim**	Sim	Sim
Button	OBJ_BUTTON	Sim	Sim	—	**Sim**	—	—
Bitmap	OBJ_BITMAP	—	Sim (apenas leitura)	Sim	—	Sim	—
Bitmap Label	OBJ_BITMAP_LABEL	Sim	Sim (apenas leitura)	—	**Sim**	Sim	—
Edit	OBJ_EDIT	Sim	Sim	—	**Sim**	—	—
Rectangle Label	OBJ_RECTANGLE_LABEL	Sim	Sim	—	**Sim**	—	—

Na tabela são utilizadas as seguintes denominações:

- **X/Y** - as coordenadas do ponto âncora são especificadas em pixels em relação a um dos cantos do gráfico;
- **Width/Height** - os objetos têm largura e altura. Se você definir "apenas leitura", isso significa que os valores para largura e altura são calculados somente após o processamento do objeto no gráfico;
- **Date/Price** - as coordenadas do ponto âncora são definidas pelo par data/preço;
- **OBJPROP_CORNER** - define o ângulo do gráfico, que especifica as coordenadas do ponto de âncora. Pode ser um dos 4 valores de enumeração ENUM_BASE_CORNER;
- **OBJPROP_ANCHOR** - define a posição do ponto de âncora no objeto, e pode ser um dos 9 valores de enumeração ENUM_ANCHOR_POINT. Precisamente a partir desse ponto até ao canto selecionado são expecificadas as cordenadas em pixels;

© 2000-2024, MetaQuotes Ltd.

- **OBJPROP_ANGLE** - define o ângulo de rotação do objeto no sentido anti-horário.

A fim de especificar o canto do gráfico, a partir do qual as coordenadas X e Y serão medidas em pixels, use ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**), onde:

- chartID - identificador do gráfico;
- name - nome de um objeto gráfico;
- OBJPROP_CORNER - propriedade ID para especificar o canto para vinculação;
- **chart_corner** - o canto desejado do gráfico, pode ser um dos valores da enumeração ENUM_BASE_CORNER.

ENUM_BASE_CORNER

ID	Descrição
CORNER_LEFT_UPPER	Centro de coordenadas está no canto superior esquerdo do gráfico
CORNER_LEFT_LOWER	Centro de coordenadas está no canto inferior esquerdo do gráfico
CORNER_RIGHT_LOWER	Centro de coordenadas está no canto inferior direito do gráfico
CORNER_RIGHT_UPPER	Centro de coordenadas está no canto superior direito do gráfico

Exemplo:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("Не удалось создать объект OBJ_LABEL ",name,", Код ошибки = ", GetLastErro
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
  }
```

© 2000-2024, MetaQuotes Ltd.

Visibilidade de Objetos

A combinação de flags de visibilidade de objeto determina as janelas de tempo de gráfico, onde o objeto é visível. Para definir/obter o valor da propriedade OBJPROP_TIMEFRAMES, você pode usar as funções ObjectSetInteger()/ObjectGetInteger().

ID	Valor	Descrição
OBJ_NO_PERIODS	0	O objeto não desenhado em nenhuma janela de tempo
OBJ_PERIOD_M1	0x00000001	O objeto é desenhado em gráficos de 1 minuto
OBJ_PERIOD_M2	0x00000002	O objeto é desenhado em gráficos de 2 minutos
OBJ_PERIOD_M3	0x00000004	O objeto é desenhado em gráficos de 3 minutos
OBJ_PERIOD_M4	0x00000008	O objeto é desenhado em gráficos de 4 minutos
OBJ_PERIOD_M5	0x00000010	O objeto é desenhado em gráficos de 5 minutos
OBJ_PERIOD_M6	0x00000020	O objeto é desenhado em gráficos de 6 minutos
OBJ_PERIOD_M10	0x00000040	O objeto é desenhado em gráficos de 10 minutos
OBJ_PERIOD_M12	0x00000080	O objeto é desenhado em gráficos de 12 minutos
OBJ_PERIOD_M15	0x00000100	O objeto é desenhado em gráficos de 15 minutos
OBJ_PERIOD_M20	0x00000200	O objeto é desenhado em gráficos de 20 minutos
OBJ_PERIOD_M30	0x00000400	O objeto é desenhado em gráficos de 30 minutos
OBJ_PERIOD_H1	0x00000800	O objeto é desenhado em gráficos de 1 hora
OBJ_PERIOD_H2	0x00001000	O objeto é desenhado em gráficos de 2 horas
OBJ_PERIOD_H3	0x00002000	O objeto é desenhado em gráficos de 3 horas
OBJ_PERIOD_H4	0x00004000	O objeto é desenhado em gráficos de 4 horas

© 2000-2024, MetaQuotes Ltd.

ID	Valor	Descrição
OBJ_PERIOD_H6	0x00008000	O objeto é desenhado em gráficos de 6 horas
OBJ_PERIOD_H8	0x00010000	O objeto é desenhado em gráficos de 8 horas
OBJ_PERIOD_H12	0x00020000	O objeto é desenhado em gráficos de 12 horas
OBJ_PERIOD_D1	0x00040000	O objeto é desenhado em gráficos diários
OBJ_PERIOD_W1	0x00080000	O objeto é desenhado em gráficos semanais
OBJ_PERIOD_MN1	0x00100000	O objeto é desenhado em gráficos mensais
OBJ_ALL_PERIODS	0x001fffff	O objeto é desenhado em todas das janelas de tempo

Os flags de visibilidade podem ser combinados usando o símbolo "|", por exemplo, a combinação de flags OBJ_PERIOD_M10|OBJ_PERIOD_H4 significa que o objeto será visível nas janelas de tempo de 10-minutos e 4-horas.

Exemplo:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // O Máxima do dia anterior
   double prevLow;            // O Mínimo do dia anterior
   double highs[],lows[];     // Arrays de Máximo e Mínimo

//--- redefine o código de erro
   ResetLastError();
//--- Obtém os 2 últimos valores de Máximo na janela de tempo diária
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // Se cópia foi bem sucedida
     {
      Print("Os preços em alta dos últimos dois dias foram obtidos com sucesso");
      prevHigh=highs[0]; // O Máximo do dia anterior
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // Objeto com o mesmo highlevel não encontrado
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // Cria o objeto de linha Horizont
        }
      //--- Define valor para nível de preço para a linha highlevel
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Define a visibilidade apenas para PERIOD_M10 e PERIOD_H4
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Não foi possível obter preços em lata nos últimos dois dias, Erro = ",Get
     }

//--- redefine o código de erro
   ResetLastError();
//--- Obtém os 2 últimos valores de Mínimo na janela de tempo diária
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // Se cópia foi bem sucedida
     {
      Print("Os preços em baixa dos últimos dois dias foram obtidos com sucesso");
      prevLow=lows[0]; // O Mínimo do dia anterior
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // Objeto com o mesmo lowlevel não encontrado
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // Cria o objeto linha Horizontal
        }
      //--- Define valor para o nível de preço para a linha lowlevel
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- Define a visibilidade apenas para PERIOD_M10 e PERIOD_H4
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Não foi possível obter preços em baixa nos últimos dois dias, Erro = ",

   ChartRedraw(0); // redesenha o gráfico de forma forçada
  }
```

Também Veja

PeriodSeconds, Period, Janela de Tempo de Gráfico, Data e Hora

© 2000-2024, MetaQuotes Ltd.

Níveis de Onda de Elliott

Ondas de Elliott são representadas por dois objetos gráficos do tipo OBJ_ELLIOTWAVE5 e OBJ_ELLIOTWAVE3. Para definir o tamanho da onda (método de rotulação de onda), a propriedade OBJPROP_DEGREE é usada, para a qual os valores da enumeração ENUM_ELLIOT_WAVE_DEGREE podem ser atribuídos.

ENUM_ELLIOT_WAVE_DEGREE

ID	Descrição
ELLIOTT_GRAND_SUPERCYCLE	Grande Super-ciclo
ELLIOTT_SUPERCYCLE	Super-ciclo
ELLIOTT_CYCLE	Ciclo
ELLIOTT_PRIMARY	Primário
ELLIOTT_INTERMEDIATE	Intermediário
ELLIOTT_MINOR	Menor
ELLIOTT_MINUTE	Minuto
ELLIOTT_MINUETTE	Minuette
ELLIOTT_SUBMINUETTE	Subminuette

Exemplo:

```
for(int i=0;i<ObjectsTotal(0);i++)
  {
   string currobj=ObjectName(0,i);
   if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
      ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
     {
      //--- define o nível de marcação para INTERMEDIÁRIO
      ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
      //--- exibe linhas entre topos de ondas
      ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
      //--- define cor de linha
      ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
      //--- define largura de linha
      ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
      //--- define descrição
      ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Objetos Gann

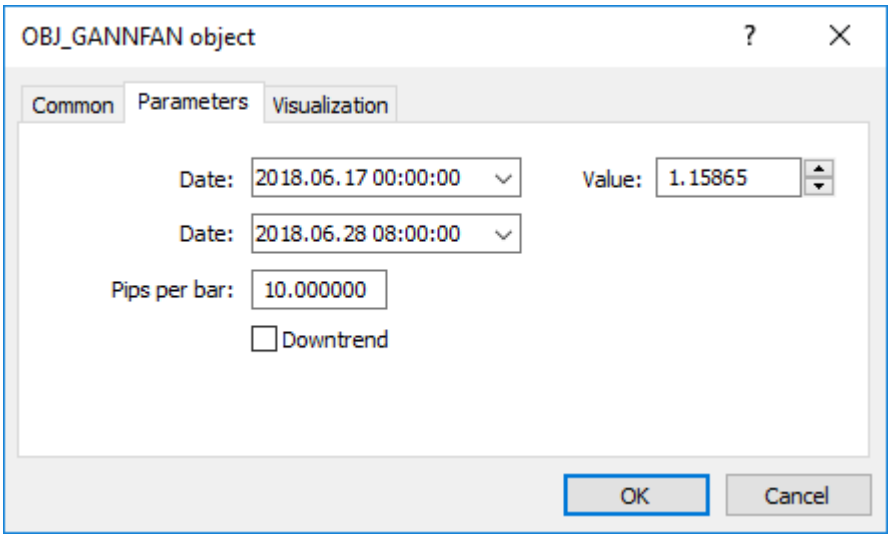
Para objetos Gann Fan (OBJ_GANNFAN) e Gann Grade (OBJ_GANNGRID) você pode especificar dois valores da enumeração ENUM_GANN_DIRECTION que definem a direção da tendência.

ENUM_GANN_DIRECTION

ID	Descrição
GANN_UP_TREND	Linha correspondente à tendência de alta
GANN_DOWN_TREND	Linha correspondente à tendência de baixa

Para definir a escala da linha principal como 1x1, use a função ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale), onde:

- chart_handle - janela do gráfico onde o objeto está localizado;
- gann_object_name - nome do objeto;
- OBJPROP_SCALE - identificador da propriedade "Escala";
- scale - escala requerida em unidades de Pips/Barra.



Exemplo de criação de um Ventilador Gann:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// Objeto não encontrado
     {
      //--- Informe sobre a falha
      Print("Objeto ",my_gann," não encontrado. Código de erro = ",GetLastError());
      //--- Obtém o preço máximo do gráfico
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- Obtém o preço mínimo do gráfico
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- Quantas barras estão exibidas no gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- Cria um array, para escrever nele a hora de abertura de cada barra
      datetime Time[];
      //--- Arruma acesso ao array como o de série de tempo
      ArraySetAsSeries(Time,true);
      //--- Agora copia dados de barras visíveis no gráfico para dentro deste array
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Não foi possível copiar o array com o tempo aberto!");
         return;
        }
      //--- Preparações preliminares concluídas

      //--- Índice da barra central no gráfico
      int center_bar=bars_on_chart/2;
      //--- Equador do gráfico - entre o máximo e mínimo
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- Define as coordenadas do primeiro ponto de ancoragem para o centro
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- Segundo ponto de ancoragem para a direita
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] = "+
      //Print("Time[center_bar]/="+Time[center_bar]+"  Time[center_bar/2]="+Time[cente
      //--- Define a escala em unidades de Pips / Barra
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- Define a tendência da linha
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- Define a largura da linha
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- Define o estilo da linha
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- Define a cor da linha
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- Permite ao usuário selecionar um objeto
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- Seleciona ele você mesmo
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- Desenha ele no gráfico
      ChartRedraw(0);
    }
  }
```

© 2000-2024, MetaQuotes Ltd.

Cores Web

As seguintes contantes de cor são definidas para o tipo color:

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

A cor pode ser definida para um objeto usando a função ObjectSetInteger(). Para definir a cor de um indicador customizado a função PlotIndexSetInteger() é usada. Para obter valores de cor existem funções similares ObjectGetInteger() e PlotIndexGetInteger().

© 2000-2024, MetaQuotes Ltd.

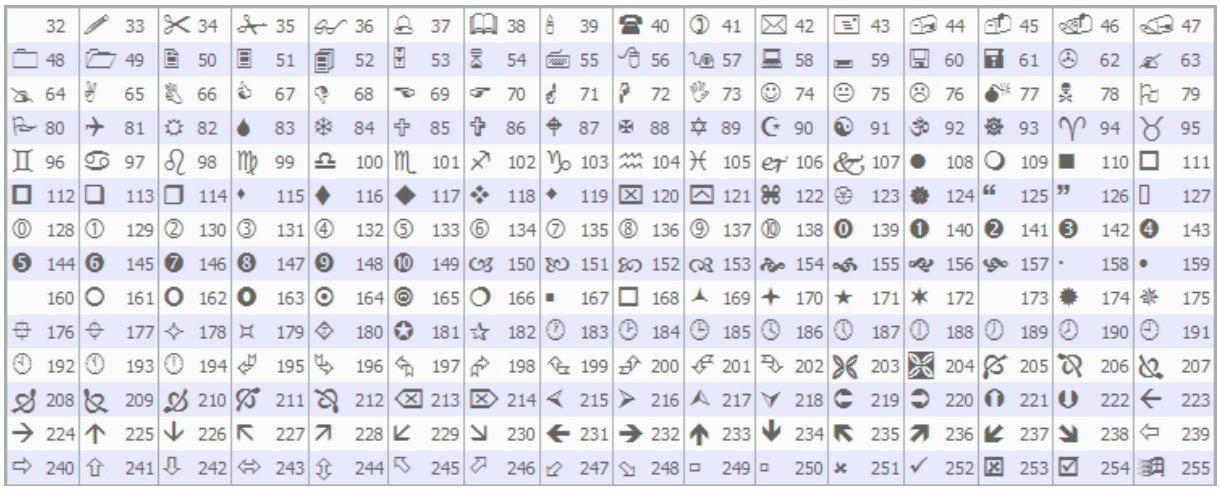
Exemplo:

```
//---- configurando um indicador
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000-2024, MetaQuotes Ltd.

Wingdings

Caracteres de Wingdings usados com o objeto OBJ_ARROW:



Um caractere necessário pode ser definido usando a função ObjectSetInteger().

Exemplo:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // Obtém o preço de Fecha
//--- Se o preço foi obtido
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // Cria uma seta
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // Define o código da set
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // Define hora
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// Define preço
      ChartRedraw(0);                                         // Desenha a seta agora
     }
   else
      Print("Não foi possível obter o último Fechamento de preço!");
  }
```

Constantes de Indicador

Existem 37 indicadores técnicos pré-definidos, que podem ser usados em programas escritos na linguagem MQL5. Além disso, existe uma oportunidade para criar indicadores customizados usando a função iCustom(). Todas as constantes requeridas para isso estão divididas em 5 grupos:

- Constantes de preço – para seleção do tipo de preço ou volume, sob o qual um indicador é calculado;
- Métodos de suavização – métodos de suavização internos usados em indicadores;
- Linhas de indicador – identificadores de buffers do indicador ao acessar valores do indicador usando CopyBuffer();
- Estilos de desenho – para indicar um dos 18 tipos de desenhos e configurações de estilo de desenho da linha;
- Propriedades de indicadores customizados são usados em funções para trabalhar com indicadores customizados;
- Tipos de indicadores são usados para especificar o tipo de indicador técnico ao criar um handle usando IndicatorCreate();
- Identificadores de tipos de dado são usados para especificar o tipo de dado passado em um array do tipo MqlParam para dentro de uma função IndicatorCreate().

© 2000-2024, MetaQuotes Ltd.

Constantes de Preço

Cálculos de indicatores técnicos requerem valores de preço e/ou valores de volumes, sob os quais cálculos serão realizados. Existem 7 indicadores pré-definidos a partir da enumeração ENUM_APPLIED_PRICE, usados para especificar o preço base desejado para os cálculos.

ENUM_APPLIED_PRICE

ID	Descrição
PRICE_CLOSE	Preço de Fechamento
PRICE_OPEN	Preço de Abertura
PRICE_HIGH	O preço máximo para o período
PRICE_LOW	O preço mínimo para o período
PRICE_MEDIAN	O preço mediano, (máximo + mínimo)/2
PRICE_TYPICAL	Preço típico, (máximo + mínimo + fechamento)/3
PRICE_WEIGHTED	Preço médio, (máximo + mínimo + fechamento + fechamento)/4

Caso o volume seja usado nos cálculos, é necessário especificar um dos dois valores a partir da enumeração ENUM_APPLIED_VOLUME.

ENUM_APPLIED_VOLUME

ID	Descrição
VOLUME_TICK	Volume de Tick
VOLUME_REAL	Volume de negociação

O indicador técnico iStochastic() pode ser calculado de duas formas usando:

- ou somente preços de Fechamento;
- ou preços de Máximo e Mínimo.

Para selecionar uma variante necessária para o cálculo, especifique um dos valores da enumeração ENUM_STO_PRICE.

ENUM_STO_PRICE

ID	Descrição
STO_LOWHIGH	Cálculo é baseado em preços Mínimo/Máximo
STO_CLOSECLOSE	Cálculo é baseado em preços Fechamento/Fechamento

Se um indicador técnico usa nos cálculos dados de preço, seu tipo é definido por ENUM_APPLIED_PRICE, então o handle de qualquer indicador (incorporado no terminal ou escrito por um usuário) pode ser usado como a série de preço de entrada. Neste caso, os valores do buffer zero

© 2000-2024, MetaQuotes Ltd.

do indicador serão utilizados para os cálculos. Isso torna fácil construir valores de um indicador usando valores de um outro indicador. O handle de um indicador customizado é criado por meio de chamada da função iCustom().

Exemplo:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- parâmetros de entrada
input int      RSIperiod=14;         // Período para cálculo do RSI
input int      Smooth=8;             // Período de suavização do RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // Método de suavização
//---- plotar RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plotar RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- buffers do indicador
double         RSIBuffer[];          // Aqui nós armazenamos os valores do RSI
double         RSI_SmoothedBuffer[]; // Aqui estarão os valores suavizados dos RSI
int            RSIhandle;            // Handle do indicador RSI
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
```

```
                  const double &price[]
                  )

  {
//---  Redefine o valor do último erro
   ResetLastError();
//--- Obtém dados do indicador RSI em um array RSIBuffer[]
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("Não é possível copiar os valores do indicador RSI. Erro = ",
            GetLastError(),",  copiado =",copied);
      return(0);
     }
//--- Cria o indicador de valores médios usando valores do RSI
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("Não é possível copiar o indicador suavizado do RSI. Erro = ",
            GetLastError(),",  copiado =",copied);
      return(0);
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Métodos de Suavização

Muitos indicadores técnicos são baseados em vários métodos de suavização de séries de preço. Alguns indicadores técnicos padrões requerem especificação da tipo de suavização como um parâmetro de entrada. Para especificar o tipo desejado de suavização, identificadores listados na enumeração ENUM_MA_METHOD são usados.

ENUM_MA_METHOD

ID	Descrição
MODE_SMA	Média simples
MODE_EMA	Média exponencial
MODE_SMMA	Média suavizada
MODE_LWMA	Média linear-ponderada

Exemplo:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles para médias móveis
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- obtém handles de MA's
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000-2024, MetaQuotes Ltd.

Linhas de Indicadores

Alguns indicadores técnicos têm vários buffers desenhados no gráfico. A numeração de buffers de um indicador começa com 0. Ao copiar valores de indicador usando a função CopyBuffer() em um array do tipo double, para alguns indicadores, pode-se indicar o identificador de um buffer copiado ao invés de seu número.

Identificadores de linhas de indicador permitidos ao copiar valores de iMACD(), iRVI() e iStochastic().

Constante	Valor	Descrição
MAIN_LINE	0	Linha principal
SIGNAL_LINE	1	Linha de sinal

Identificadores de linhas de indicador permitidos ao copiar valores de ADX() e ADXW().

Constante	Valor	Descrição
MAIN_LINE	0	Linha principal
PLUSDI_LINE	1	Linha +DI
MINUSDI_LINE	2	Linha -DI

Identificadores de linhas de indicador permitidos ao copiar valores de iBands().

Constante	Valor	Descrição
BASE_LINE	0	Linha principal
UPPER_BAND	1	Limite superior
LOWER_BAND	2	Limite inferior

Identificadores de linhas de indicador permitidos ao copiar valores de iEnvelopes() e iFractals().

Constante	Valor	Descrição
UPPER_LINE	0	Linha de cima
LOWER_LINE	1	Linha de baixo

Identificadores de linhas de indicador permitidos ao copiar valores de iGator()

Constante	Valor	Descrição
UPPER_HISTOGRAM	0	Histograma superior
LOWER_HISTOGRAM	2	Histograma inferior

Identificadores de linhas de indicador permitidos ao copiar valores de iAlligator().

© 2000-2024, MetaQuotes Ltd.

Constante	Valor	Descrição
GATORJAW_LINE	0	Linha Jaw
GATORTEETH_LINE	1	Linha Teeth
GATORLIPS_LINE	2	Linha Lips

Identificadores de linhas de indicador permitidos ao copiar valores de iIchimoku().

Constante	Valor	Descrição
TENKANSEN_LINE	0	Linha Tenkan-sen
KIJUNSEN_LINE	1	Linha Kijun-sen
SENKOUSPANA_LINE	2	Linha Senkou Span A
SENKOUSPANB_LINE	3	Linha Senkou Span B
CHIKOUSPAN_LINE	4	Linha Chikou Span

© 2000-2024, MetaQuotes Ltd.

Estilos de Desenho

Ao criarum indicador customizado, pode-se especificar um dos 18 tipos de plotagem gráfica (como exibidos na janela do gráfico principal ou em uma sub-janela do gráfico), cujos valores são especificados na enumeração ENUM_DRAW_TYPE.

Em um indicador customizado, é permitido usar quaisquer tipos de indicador construído/desenho. Cada tipo de construção requer especificação de um a cinco arrays globais para armazenar dados necessários para o desenho. Estes arrays de dados devem ser associados com buffers de indicador usando a função SetIndexBuffer(). O tipo de dados de ENUM_INDEXBUFFER_TYPE deve ser especificado para cada buffer.

Dependendo do estilo de desenho, pode-se necessitar de um a quatro buffers de valor (marcados como INDICADOR_DATA). Se um estilo admite alternação dinâmica de cores (todos os estilos contém COLOR nos seus nomes), então você precisará de um buffer adicional de cor (tipo indicado INDICATOR_COLOR_INDEX). Os buffers de cor são sempre vinculados após os buffers de valores correspondendo ao estilo.

ENUM_DRAW_TYPE

ID	Descrição	Estruturas de Dados	Buffers de cor
DRAW_NONE	Não desenhado	1	0
DRAW_LINE	Linha	1	0
DRAW_SECTION	Seção	1	0
DRAW_HISTOGRAM	Histograma a partir da linha zero	1	0
DRAW_HISTOGRAM2	Histograma de dois buffers do indicador	2	0
DRAW_ARROW	Desenha setas	1	0
DRAW_ZIGZAG	Estilo Zigzag permite divisão vertical na barra	2	0

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Estruturas de Dados	Buffers de cor
DRAW_FILLING	Preenchimento de cor entre dos dois níveis	2	0
DRAW_BARS	Exibe como uma seqüência de barras	4	0
DRAW_CANDLES	Exibe como um seqüência de candles	4	0
DRAW_COLOR_LINE	Linha Multicolorida	1	1
DRAW_COLOR_SECTION	Divisão Multicolorida	1	1
DRAW_COLOR_HISTOGRAM	Histograma multicolorida a partir da linha zero	1	1
DRAW_COLOR_HISTOGRAM2	Histograma multicolorido dos dois buffers do indicador	2	1
DRAW_COLOR_ARROW	Desenho de setas multicoloridas	1	1
DRAW_COLOR_ZIGZAG	ZigZag Multicolo	2	1

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Estruturas de Dados	Buffers de cor
	rido		
DRAW_COLOR_BARS	Barras multicoloridas	4	1
DRAW_COLOR_CANDLES	Candles multicoloridos	4	1

Para refinar a exibição dos tipos de desenho selecionados, identificadores listados em ENUM_PLOT_PROPERTY são usados.

Para funções PlotIndexSetInteger() e PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

ID	Descrição	Tipo da Propriedade
PLOT_ARROW	Código de seta para estilo DRAW_ARROW	uchar
PLOT_ARROW_SHIFT	Deslocamento vertical de setas para estilo DRAW_ARROW	int
PLOT_DRAW_BEGIN	Número de barras iniciais sem desenho e valores na Janela de Dados	int
PLOT_DRAW_TYPE	Tipo de construção gráfica	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Sinal de exibição de valores de construção na Janela de Dados	bool
PLOT_SHIFT	Deslocamento da plotagem do indicador ao longo do eixo tempo em barras	int
PLOT_LINE_STYLE	Estilo de linha de desenho	ENUM_LINE_STYLE

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo da Propriedade
PLOT_LINE_WIDTH	A espessura da linha de desenho	int
PLOT_COLOR_INDEXES	O número de cores	int
PLOT_LINE_COLOR	O índice de um buffer contendo a cor de desenho	color modificador = número de índice de cores

Para a função PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

ID	Descrição	Tipo da Propriedade
PLOT_EMPTY_VALUE	Um valor vazio para plotagem, para a qual não há desenho	double

Para a função PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

ID	Descrição	Tipo da Propriedade
PLOT_LABEL	O nome da série gráfica do indicador para exibir na Janela de Dados Ao trabalhar com estilos gráficos complexos exigindo vários buffers de indicador para a exposição, os nomes para cada buffer pode ser especificado utilizando ";" como separador. O código de exemplo é mostrado na DRAW_CANDLES	string

5 estilos podem ser usados para desenhar linhas em indicadores customizados. Eles são válidos somente para a espessura de linha 0 ou 1.

© 2000-2024, MetaQuotes Ltd.

ENUM_LINE_STYLE

ID	Descrição
STYLE_SOLID	Linha sólida
STYLE_DASH	Linha tracejada
STYLE_DOT	Linha pontilhada
STYLE_DASHDOT	Linha traço-ponto
STYLE_DASHDOTDOT	Traço - dois pontos

Para definir o estilo de desenho de linha e o tipo de desenho, a função PlotIndexSetInteger()é usada. Para as extensões de Fibonacci, a espessura e o estilo de desenho dos níveis podem ser indicados usando a função ObjectSetInteger().

Exemplo:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- buffers do indicador
double         MABuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Associa o array ao buffer do indicador com índice 0
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- Define o desenho da linha
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- Define o estilo da linha
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- Define cor de linha
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- Define espessura de linha
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- Define rótulos para a linha
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

© 2000-2024, MetaQuotes Ltd.

```
                  const double &high[],
                  const double &low[],
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Propriedades de Indicador Customizado

O número de buffers de indicador que podem ser usados em um indicador customizado é ilimitado. Mas para cada array que é designado como um buffer de indicador usando a função SetIndexBuffer(), é necessário especificar o tipo de dado que ele armazenará. Este pode ser um dos valores da enumeração ENUM_INDEXBUFFER_TYPE.

ENUM_INDEXBUFFER_TYPE

ID	Descrição
INDICATOR_DATA	Dados para desenho
INDICATOR_COLOR_INDEX	Cor
INDICATOR_CALCULATIONS	Buffers auxiliares para cálculos intermediários

Um indicador customizado tem muitas configurações para fornecer uma exibição conveniente. Estas configurações são feitas através da atribuição de propriedades de indicador correspondentes usando a funções IndicatorSetDouble(), IndicatorSetInteger() e IndicatorSetString(). Identificadores de propriedades de indicador são listados na enumeração ENUM_CUSTOMIND_PROPERTY.

ENUM_CUSTOMIND_PROPERTY_INTEGER

ID	Descrição	Tipo da Propriedade
INDICATOR_DIGITS	Acuracidade de desenho de valores de indicador	int
INDICATOR_HEIGHT	Altura fixa da janela do indicador (o	int

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo da Propriedade
	comando de pré-processador #property indicator_height)	
INDICATOR_LEVELS	Número de níveis na janela do indicador	int
INDICATOR_LEVELCOLOR	Cor da linha de nível	color modificador = número de nível
INDICATOR_LEVELSTYLE	Estilo da linha de nível	ENUM_LINE_STYLE modificador = número de nível
INDICATOR_LEVELWIDTH	Espessura da linha de	int modificador = número de nível

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo da Propriedade
	nível	
INDICATOR_FIXED_MINIMUM	Mínimo fixo para a janela do indicador. A propriedade só pode ser gravada pela função IndicatorSetInteger()	bool
INDICATOR_FIXED_MAXIMUM	Máximo fixo para a janela do indicador. A propried	bool

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo da Propriedade
	ade só pode ser gravada pela função IndicatorSetInteger()	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

ID	Descrição	Tipo da Propriedade
INDICATOR_MINIMUM	Mínimo da janela do indicador	double
INDICATOR_MAXIMUM	Máximo da janela do indicador	double
INDICATOR_LEVELVALUE	Valor de nível	double modificador = número de nível

ENUM_CUSTOMIND_PROPERTY_STRING

© 2000-2024, MetaQuotes Ltd.

ID	Descrição	Tipo da Propriedade
INDICATOR_SHORTNAME	Nome do indicador abreviado	string
INDICATOR_LEVELTEXT	Nível de descrição	string modificador = número de nível

Exemplos:

```
//--- configurações do indicador
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- parâmetros de entrada
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- buffers do indicador
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- define acuracidade
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- define níveis
```

© 2000-2024, MetaQuotes Ltd.

```
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- define máximo e mínimo para sub-janela
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- define a primeira barra a partir do qual o índice será desenhado
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- define estilo STYLE_DOT para a segunda linha
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- nome para Janela de Dados e rótulo da sub-janela do indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- define valor vazio para linha de desenho
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- inicialização concluída
  }
```

© 2000-2024, MetaQuotes Ltd.

Indicadores Técnicos

Existem duas formas de criar via programação um indicador para posteriormente acessar seus valores. A primeira forma é especificar diretamente um nome de função da lista de indicadores técnicos. A segunda forma é, usando a função IndicatorCreate(), criar de forma uniforme um handle de qualquer indicador através da atribuição de um identificador da enumeração ENUM_INDICATOR. Ambos as formas de criar handles são iguais, você pode usar aquela que for mais conveniente em um caso particular ao escrever um programa em MQL5.

Ao criar um indicador do tipo IND_CUSTOM, o campo *type* do primeiro elemento de um array de parâmetros de entrada MqlParam deve ter valor TYPE_STRING da enumeração ENUM_DATATYPE, enquanto que o campo *string_value* do primeiro elemento deve conter o nome do indicador customizado.

ENUM_INDICATOR

Identificador	Indicador
IND_AC	Accelerator Oscillator
IND_AD	Accumulation/Distribution
IND_ADX	Average Directional Index
IND_ADXW	ADX by Welles Wilder
IND_ALLIGATOR	Alligator
IND_AMA	Adaptive Moving Average
IND_AO	Awesome Oscillator
IND_ATR	Average True Range
IND_BANDS	Bollinger Bands®
IND_BEARS	Bears Power
IND_BULLS	Bulls Power
IND_BWMFI	Market Facilitation Index
IND_CCI	Commodity Channel Index
IND_CHAIKIN	Chaikin Oscillator
IND_CUSTOM	Custom indicator
IND_DEMA	Double Exponential Moving Average
IND_DEMARKER	DeMarker
IND_ENVELOPES	Envelopes
IND_FORCE	Force Index
IND_FRACTALS	Fractals
IND_FRAMA	Fractal Adaptive Moving Average

© 2000-2024, MetaQuotes Ltd.

Identificador	Indicador
IND_GATOR	Gator Oscillator
IND_ICHIMOKU	Ichimoku Kinko Hyo
IND_MA	Moving Average
IND_MACD	MACD
IND_MFI	Money Flow Index
IND_MOMENTUM	Momentum
IND_OBV	On Balance Volume
IND_OSMA	OsMA
IND_RSI	Relative Strength Index
IND_RVI	Relative Vigor Index
IND_SAR	Parabolic SAR
IND_STDDEV	Standard Deviation
IND_STOCHASTIC	Stochastic Oscillator
IND_TEMA	Triple Exponential Moving Average
IND_TRIX	Triple Exponential Moving Averages Oscillator
IND_VIDYA	Variable Index Dynamic Average
IND_VOLUMES	Volumes
IND_WPR	Williams' Percent Range

© 2000-2024, MetaQuotes Ltd.

Identificadores de Tipo de Dados

Quando criar um indicador usando a função IndicatorCreate(), um array de tipo MqlParam deve ser especificado como o último parâmetro. Por sua vez, a estrutura MqlParam, descrevendo o indicador, contém um *tipo* de campo especial. Este campo contém informação sobre o tipo de dado (real, integer pi string type) que é passado por um elemento específico do array. O valor deste campo da estrutura MqlParam pode ser um dos valores ENUM_DATATYPE.

ENUM_DATATYPE

Identificador	Tipo de dado
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	color
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

Cada elemento do array descreve o correspondente parâmetro de entrada de um indicador técnico criado, assim o tipo e a ordem dos elementos no array devem ser rigorosamente mantidos em concordância com a descrição.

© 2000-2024, MetaQuotes Ltd.

Estado de Ambiente

As constantes que descrevem o ambiente de execução corrente de um programa MQL5 estão divididas em grupos:

- Propriedades do terminal cliente – informações sobre o terminal cliente;
- Propriedades do programa-MQL5 executado – propriedades do programa MQL5, que ajuda a controlar sua execução;
- Propriedades do ativo – obtenção de informações sobre um ativo;
- Propriedades da conta – informações sobre a conta atual;
- Estatísticas de teste – resultados de teste de um Expert Advisor.

© 2000-2024, MetaQuotes Ltd.

Propriedades do Terminal Cliente

Informações sobre o terminal cliente podem ser obtidos através de duas funções: TerminalInfoInteger() e TerminalInfoString(). Por parâmetros, estas funções aceitam valores do ENUM_TERMINAL_INFO_INTEGER e ENUM_TERMINAL_INFO_STRING respectivamente.

ENUM_TERMINAL_INFO_INTEGER

Identificador	Descrição	Tipo
TERMINAL_BUILD	O número de compilação (build) do terminal cliente	int
TERMINAL_COMMUNITY_ACCOUNT	A flag indica a presença de dados autorizados do MQL5.community no terminal	int
TERMINAL_COMMUNITY_CONNECTION	Conexão com MQL5.community	bool
TERMINAL_CONNECTED	Conexão com um servidor de negociação	bool
TERMINAL_DLLS_ALLOWED	Permissão para usar DLL	bool
TERMINAL_TRADE_ALLOWED	Permissão para negociar	bool
TERMINAL_EMAIL_ENABLED	Permissão para enviar e-mails usando servidor-SMTP e login, especificado nas configurações do terminal	bool
TERMINAL_FTP_ENABLED	Permissão para enviar relatórios usando servidor-FTP e login, especificado nas configurações do terminal	bool
TERMINAL_NOTIFICATIONS_ENABLED	Permissão para enviar notificações ao smartphone	bool
TERMINAL_MAXBARS	A quantidade máxima de barras no gráfico	int
TERMINAL_MQID	O sinalizador indica a presença de dados de ID do MetaQuotes para enviar notificações Push	bool
TERMINAL_CODEPAGE	Número do código de página da linguagem instalado no terminal cliente	int
TERMINAL_CPU_CORES	O número de núcleos de CPU no sistema	int
TERMINAL_DISK_SPACE	Espaço livre de disco para a pasta MQL5\Files do terminal (agente), MB	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
TERMINAL_MEMORY_PHYSICAL	Memória física no sistema, MB	int
TERMINAL_MEMORY_TOTAL	Memória disponível para o processo do terminal (agente), MB	int
TERMINAL_MEMORY_AVAILABLE	Memória livre do processo do terminal (agente), MB	int
TERMINAL_MEMORY_USED	Memória usada pelo terminal (agente), MB	int
TERMINAL_X64	Indicação do "terminal 64-bit"	bool
TERMINAL_OPENCL_SUPPORT	A versão do OpenCL suportado no formato 0x00010002 = 1.2. "0" significa que OpenCL não é suportado	int
TERMINAL_SCREEN_DPI	A qualidade de resolução ao exibir informações no ecrã é medida pelo número de pontos por polegada linear da superfície (DPI). O conhecimento desse parâmetro permite que você especifique as dimensões dos objetos gráficos de modo que pareçam iguais em monitores com uma resolução diferente.	int
TERMINAL_SCREEN_LEFT	Coordenada esquerda da tela virtual. A tela virtual é o retângulo que cobre todos os monitores. Se o sistema tiver dois monitores e sua ordem for definida da direita para a esquerda, a coordenada esquerda da tela virtual poderá estar na borda de dois monitores.	int
TERMINAL_SCREEN_TOP	Coordenada superior da tela virtual	int
TERMINAL_SCREEN_WIDTH	Largura do terminal	int
TERMINAL_SCREEN_HEIGHT	Altura do terminal	int
TERMINAL_LEFT	Coordenada esquerda do terminal em relação à tela virtual	int
TERMINAL_TOP	Coordenada direita do terminal em relação à tela virtual	int
TERMINAL_RIGHT	Coordenada superior do terminal em relação à tela virtual	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
TERMINAL_BOTTOM	Coordenada inferior do terminal em relação à tela virtual	int
TERMINAL_PING_LAST	Último valor conhecido do ping até ao servidor de negociação em microssegundos. Em um segundo existe um milhão de microssegundos	int
TERMINAL_VPS	Sinal de que o terminal está sendo executado no servidor virtual MetaTrader Vitrtual Hosting (VPS MetaTrader)	bool
Identificador de tecla	**Descrição**	
TERMINAL_KEYSTATE_LEFT	Status da tecla "Seta para a esquerda"	int
TERMINAL_KEYSTATE_UP	Status da tecla "Seta para cima"	int
TERMINAL_KEYSTATE_RIGHT	Status da tecla "Seta para a direita"	int
TERMINAL_KEYSTATE_DOWN	Status da tecla "Seta para baixo"	int
TERMINAL_KEYSTATE_SHIFT	Status da tecla "Shift"	int
TERMINAL_KEYSTATE_CONTROL	Status da tecla "Ctrl"	int
TERMINAL_KEYSTATE_MENU	Status da tecla "Windows"	int
TERMINAL_KEYSTATE_CAPSLOCK	Status da tecla "CapsLock"	int
TERMINAL_KEYSTATE_NUMLOCK	Status da tecla "NumLock"	int
TERMINAL_KEYSTATE_SCRLOCK	Status da tecla "ScrollLock"	int
TERMINAL_KEYSTATE_ENTER	Status da tecla "Enter"	int
TERMINAL_KEYSTATE_INSERT	Status da tecla "Insert"	int
TERMINAL_KEYSTATE_DELETE	Status da tecla "Delete"	int
TERMINAL_KEYSTATE_HOME	Status da tecla "Home"	int
TERMINAL_KEYSTATE_END	Status da tecla "End"	int
TERMINAL_KEYSTATE_TAB	Status da tecla "Tab"	int
TERMINAL_KEYSTATE_PAGEUP	Status da tecla "PageUp"	int
TERMINAL_KEYSTATE_PAGEDOWN	Status da tecla "PageDown"	int
TERMINAL_KEYSTATE_ESCAPE	Status da tecla "Escape"	int

A chamada de TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) retorna o mesmo código de status de tecla como a função GetKeyState() a partir de MSDN.

© 2000-2024, MetaQuotes Ltd.

Exemplo de como calcular o coeficiente da escala:

```
//--- criamos um botão de 1.5 polegadas de largura no ecrã
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // obtemos o DPI do monitor
int base_width = 144;                                      // largura básica em pixels
int width      = (button_width * screen_dpi) / 96;         // calculamos a largura do
...

//--- cálculo do coeficiente da escala em porcentagens
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- uso do coeficiente de escala
width=(base_width * scale_factor) / 100;
```

Usando o recurso gráfico desse modo, ele irá ter o mesmo tamanho, à primeira vista, em monitores de diferente qualidade de resolução. Além disso, os parâmetros que controlam os elementos (botões, caixas de diálogo, etc.) irão corresponder às configurações personalizadas.

ENUM_TERMINAL_INFO_DOUBLE

Identifier	Descrição	tipo
TERMINAL_COMMUNITY_BALANCE	Saldo no MQL5.community	double
TERMINAL_RETRANSMISSION	Porcentagem de pacotes de rede re-enviados no protocolo TCP/IP para todos os aplicativos e serviços executados no computador. A perda de pacotes ocorre mesmo em redes rápidas e configuradas corretamente. Quando ela ocorre, não há confirmação da entrega de pacotes entre o receptor e o remetente. Por conseguinte, pacotes "perdidos" são reenviados. Ele não é uma indicação da qualidade da conexão entre um terminal particular e um servidor de negociação, uma vez que a porcentagem é calculada para toda a atividade da rede, incluindo a de sistema e de fundo. O valor TERMINAL_TRESSMISSION é solicitado no sistema operacional uma vez por minuto. O próprio terminal não calcula estes valores.	double

© 2000-2024, MetaQuotes Ltd.

Operações de arquivo podem ser realizadas somente em dois diretórios; os caminhos correspondentes podem ser obtidos usando solicitação para as propriedades TERMINAL_DATA_PATH e TERMINAL_COMMONDATA_PATH.

ENUM_TERMINAL_INFO_STRING

Identificador	Descrição	Tipo
TERMINAL_LANGUAGE	Linguagem do terminal	string
TERMINAL_COMPANY	Nome da empresa	string
TERMINAL_NAME	Nome do terminal	string
TERMINAL_PATH	Pasta a partir da qual o terminal foi iniciado	string
TERMINAL_DATA_PATH	Pasta na qual dados do terminal são armazenados	string
TERMINAL_COMMONDATA_PATH	Caminho comum para todos os terminais instalados em um computador	string
TERMINAL_CPU_NAME	Nome do processador	string
TERMINAL_CPU_ARCHITECTURE	Arquitetura do processador	string
TERMINAL_OS_VERSION	Nome do sistema operacional do usuário	string

Para um melhor entendimento dos caminhos armazenados nas propriedades dos parâmetros TERMINAL_PATH, TERMINAL_DATA_PATH e TERMINAL_COMMONDATA_PATH, é recomendado executar o script, que retornará estes valores para a cópia corrente do terminal cliente, instalado em seu computador.

Example: Script retorna informações sobre os caminhos do terminal cliente

```
//+----------´-------------------------------------------------------+
//|                                          Check_TerminalPaths.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+----------´-------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+----------´-------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+----------´-------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
  Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
}
```

Como resultado da execução do script no Diário de Experts, você verá mensagens, como as seguintes:

Toolbox

Time	Source	Message
2018.06.29 09:28:27.253	Paths (EURUSD,H1)	TERMINAL_PATH = C:\Program Files\MetaTrader 5
2018.06.29 09:28:27.254	Paths (EURUSD,H1)	TERMINAL_DATA_PATH = C:\Users\smith\AppData\Roaming\MetaQuotes\...
2018.06.29 09:28:27.254	Paths (EURUSD,H1)	TERMINAL_COMMONDATA_PATH = C:\Users\smith\AppData\Roaming\MetaQ...

Trade | Exposure | History | News | Mailbox 7 | Calendar | Company | Market | Alerts | Signals | Code Base | Experts

© 2000-2024, MetaQuotes Ltd.

Propriedades do Programa MQL5 em Execução

Para obter informações sobre o programa mql5 em execução no momento, as constantes ENUM_MQL_INFO_INTEGER e ENUM_MQL_INFO_STRING são usadas.

Para a função MQLInfoInteger

ENUM_MQL_INFO_INTEGER

Identificador	Descrição	Tipo
MQL_HANDLES_USED	Número atual de identificadores de objetos ativos. São considerados tanto objetos dinâmicos (criados por meio de new), bem como objetos não dinâmicos, variáveis globais/locais ou membros de classe. Quanto mais identificadores um programa usa, mais recursos ele consome.	int
MQL_MEMORY_LIMIT	Quantidade máxima possível de memória dinâmica para o programa MQL5 em MB	int
MQL_MEMORY_USED	O tamanho de memória usado pelo programa MQL5 em MB	int
MQL_PROGRAM_TYPE	Tipo do programa MQL5	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	A permissão para usar DLL **para o dado programa executado**	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
MQL_TRADE_ALLOWED	A permissão para negociar **para o dado programa executado**	bool
MQL_SIGNALS_ALLOWED	A permissão para modificar os Sinais **para o determinado programa executado**	bool
MQL_DEBUG	O flag, que indica o modo de depuração	bool
MQL_PROFILER	O flag, que indica que o programa está operando em modo de profiling de código	bool
MQL_TESTER	O flag, que indica o processo do testador	bool
MQL_FORWARD	O flag que indica o processo de otimização para frente (forward testing)	bool
MQL_OPTIMIZATION	O flag, que indica o processo de otimização	bool
MQL_VISUAL_MODE	O flag, que indica o processo de testador visual	bool
MQL_FRAME_MODE	O sinalizador, indicando que o Expert Advisor está operando na coleta de resultado no modo de otimização dos quadros	bool
MQL_LICENSE_TYPE	Tipo de licença do módulo EX5. A licença refere-se	ENUM_LICENSE_TYPE

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ao módulo EX5, a partir do qual é feita uma solicitação usando MQLInfoInteger(MQL_LICENSE_TYPE).	

Para a função MQLInfoString

ENUM_MQL_INFO_STRING

Identificador	Descrição	Tipo
MQL_PROGRAM_NAME	Nome do programa MQL5 executado	string
MQL_PROGRAM_PATH	Caminho **para o dado programa executado**	string

Para obter informações sobre o tipo do programa em execução, são utilizados valores de ENUM_PROGRAM_TYPE.

ENUM_PROGRAM_TYPE

Identificador	Descrição
PROGRAM_SCRIPT	Script
PROGRAM_EXPERT	Expert
PROGRAM_INDICATOR	Indicador
PROGRAM_SERVICE	Serviço

ENUM_LICENSE_TYPE

Identificador	Descrição
LICENSE_FREE	Um versão gratuita ilimitada
LICENSE_DEMO	Um versão trial de um produto pago do Market. Funciona somente no Provador de Estratégia
LICENSE_FULL	Uma versão licenciada adquirida permite pelo menos 5 ativações. O vendedor pode aumentar o número permitido de ativações.

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
LICENSE_TIME	Uma versão com termo de licença limitada

Exemplo:

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
  {
   case PROGRAM_SCRIPT:
     {
      Print(__FILE__+" is script");
      break;
     }
   case PROGRAM_EXPERT:
     {
      Print(__FILE__+" is Expert Advisor");
      break;
     }
   case PROGRAM_INDICATOR:
     {
      Print(__FILE__+" is custom indicator");
      break;
     }
   default:Print("MQL5 type value is ",mql_program);
//---
Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), " MB")
Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), " MB");
Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), " hand
```

© 2000-2024, MetaQuotes Ltd.

Propriedades do Ativo

Para obter as informações atuais do mercado, existem várias funções: SymbolInfoInteger(), SymbolInfoDouble() e SymbolInfoString(). O primeiro parâmetro é o nome do ativo, os valores do segundo parâmetro da função podem ser um dos identificadores ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE e ENUM_SYMBOL_INFO_STRING.

Para a função SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

Identificador	Descrição	Tipo
SYMBOL_SUBSCRIPTION_DELAY	Indica que os dados do símbolo sofreram atraso. A propriedade só pode ser consultada para símbolos selecionados em Market Wat	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ch (SYMBOL_SELECT = true). Um erro será gerado para símbolos não selecionados ERR_MARKET_NOT_SELECTED (4302)	
SYMBOL_SECTOR	O setor da economia ao qual pertence o ativo	ENUM_SYMBOL_SECTOR

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
SYMBOL_INDUSTRY	Tipo de indústria ou ramo da economia ao qual pertence o ativo	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	Indica que o ativo é personalizado, em outras palavras, criado artificialmente com base em outros símbolo	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	s da janela Observação de mercado ou/e fontes de dados externas	
SYMBOL_BACKGROUND_COLOR	Cor de fundo que destaca o um símbolo na Observação de mercado	color
SYMBOL_CHART_MODE	Tipo de preço para construção	ENUM_SYMBOL_CHART_MODE

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de barras - Bid ou Last	
SYMBOL_EXIST	Sinal de que existe um símbolo com o mesmo nome	bool
SYMBOL_SELECT	O ativo (símbolo) é selecionado na janela Observação de Mercado	bool
SYMBOL_VISIBLE	Indica que o símbolo	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	selecionado é exibido na Observação de Mercado. Alguns símbolos (normalmente, taxas transversais necessárias para o cálculo de requisitos de margem e lucro na moeda	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	depósito) são selecionados automaticamente, mas, neste caso, não podem ser mostrados na Observação de Mercado. Para serem exibidos, esses símbolos devem	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ser selecionados explicitamente.	
SYMBOL_SESSION_DEALS	Número de operações (deals) na sessão atual	long
SYMBOL_SESSION_BUY_ORDERS	Número de ordens de Compra no momento	long
SYMBOL_SESSION_SELL_ORDERS	Número de ordens de Venda no mo	long

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	mento	
SYMBOL_VOLUME	Volume da última operação (deal)	long
SYMBOL_VOLUMEHIGH	Volume diário máximo	long
SYMBOL_VOLUMELOW	Volume diário mínimo	long
SYMBOL_TIME	Hora da última cotação	datetime
SYMBOL_DIGITS	Dígitos após um ponto decimal	int
SYMBOL_SPREAD_FLOAT	Indicação de um spre	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ad flutuante	
SYMBOL_SPREAD	Valor do spread em pontos	int
SYMBOL_TICKS_BOOKDEPTH	Número máximo de solicitações exibidas na Profundidade do Mercado. Para ativos que não tem fila de solicitações, o valor é igual a	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	zero.	
SYMBOL_TRADE_CALC_MODE	Modo de cálculo de preço de contrato	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	Tipo de execução de ordem	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	Data do início de negociação do ativo (geralmente usado para futuros)	datetime
SYMBOL_EXPIRATION_TIME	Data do fim de negociação do ativ	datetime

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	o (geralmente usado para futuros)	
SYMBOL_TRADE_STOPS_LEVEL	Mínima distância em pontos a partir do preço de fechamento atual para colocar ordens de Stop	int
SYMBOL_TRADE_FREEZE_LEVEL	Distância para congelar operações de neg	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ociação em pontos	
SYMBOL_TRADE_EXEMODE	Modo de execução de operação (deal)	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	Modelo de cálculo de Swap	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	Dia da semana para cobrar rolagem de swap de 3 dias	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	Modo de cálculo de margem de	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	cobertura de acordo com o lado maior - leg - (Buy ou Sell)	
SYMBOL_EXPIRATION_MODE	Flags de modos de expiração de ordens permitidas	int
SYMBOL_FILLING_MODE	Flags de modos de preenchimento de ordens permitidas	int
SYMBOL_ORDER_MODE	Flags de	int

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	modos de preenchimento de ordens permitidas	
SYMBOL_ORDER_GTC_MODE	Prazo de validade do StopLoss e TakeProfit das ordens, se SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC (Good till canc	ENUM_SYMBOL_ORDER_GTC_MODE

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	elled)	
SYMBOL_OPTION_MODE	Tipo de opção	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Opção direita (Call/Put)	ENUM_SYMBOL_OPTION_MODE

Para a função SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

Identificador	Descrição	Tipo
SYMBOL_BID	Preço de Venda (Bid) - melhor oferta de venda	double
SYMBOL_BIDHIGH	Máximo Preço de Venda do dia	double
SYMBOL_BIDLOW	Mínimo Preço de Venda do dia	double
SYMBOL_ASK	Preço de Compra (Ask) - melhor oferta	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de compra	
SYMBOL_ASKHIGH	Máximo Preço de Compra do dia	double
SYMBOL_ASKLOW	Mínimo Preço de Compra do dia	double
SYMBOL_LAST	Preço da último operação (deal)	double
SYMBOL_LASTHIGH	Máximo dos preços negociados do dia	double
SYMBOL_LASTLOW	Mínimo dos preços negociados do dia	double
SYMBOL_VOLUME_REAL	Volume da última operação (deal)	double
SYMBOL_VOLUMEHIGH_REAL	Volume diário máximo	double
SYMBOL_VOLUMELOW_REAL	Volume diário mínimo	double
SYMBOL_OPTION_STRIKE	O preço do strike	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de uma opção. O preço que um comprador de opções pode comprar (numa opção de compra) ou vender (numa opção de venda) o ativo subjacente e o vendedor de opções é obrigado a vender ou a comprar a quantia adequada do ativo subjacente.	
SYMBOL_POINT	Valor de um ponto do ativo	double
SYMBOL_TRADE_TICK_VALUE	Valor do SYMBOL_TRADE_TICK_V	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ALUE_PROFIT	
SYMBOL_TRADE_TICK_VALUE_PROFIT	Preço de tick calculado para uma posição lucrativa	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Preço de tick calculado para um posição perdedora	double
SYMBOL_TRADE_TICK_SIZE	Mínima mudança de preço	double
SYMBOL_TRADE_CONTRACT_SIZE	Tamanho do contrato de negociação	double
SYMBOL_TRADE_ACCRUED_INTEREST	Os Juros acumuladossão uma parte dos juros dos títulos e é calculado proporcionalmente ao número	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de dias a partir da data de emissão do cupom ou da data de pagamento do juro anterior.	
SYMBOL_TRADE_FACE_VALUE	O valor nominal é o valor inicial - de um título - definido pelo emissor	double
SYMBOL_TRADE_LIQUIDITY_RATE	O índice de liquidez é a proporção do valor do ativo que pode ser usada como garantia.	double
SYMBOL_VOLUME_MIN	Mínimo volume para uma operação (deal)	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
SYMBOL_VOLUME_MAX	Máximo volume para uma operação (deal)	double
SYMBOL_VOLUME_STEP	Mínimo passo de mudança de volume para execução de uma operação (deal)	double
SYMBOL_VOLUME_LIMIT	Máximo volume agregado permitido de uma posição aberta e ordens pendentes em uma direção (compra ou venda) para um ativo. Por exemplo, com a limitação de 5 lotes, você pode	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	ter uma posição comprada aberta com o volume de 5 lotes e colocar uma ordem pendente de Sell Limit com o volume de 5 lotes. Mas neste caso você não pode colocar um ordem pendente de Buy Limite (já que o total de volume em uma direção excederia o limite) ou colocar Sell Limit com o volume	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	maior que 5 lotes.	
SYMBOL_SWAP_LONG	Valor de swap comprado	double
SYMBOL_SWAP_SHORT	Valor de swap vendido	double
SYMBOL_SWAP_SUNDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir uma posição de um determinado - para (SUNDAY) - para o próximo. Podem-se aplicar os seguintes valores: • 0 - não	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	os são acumulados swaps • 1 - único swap acumulado • 3 - triplo	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	swapacumulado	
SYMBOL_SWAP_MONDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir a posição de segunda para terça-feira	double
SYMBOL_SWAP_TUESDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	transferir a posição de terça para quarta-feira	
SYMBOL_SWAP_WEDNESDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir a posição de quarta para quinta-feira	double
SYMBOL_SWAP_THURSDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir a posição de quinta para	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	sexta-feira	
SYMBOL_SWAP_FRIDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir a posição de sexta para sábado	double
SYMBOL_SWAP_SATURDAY	Coeficiente de acumulação de swaps (SYMBOL_SWAP_LONG ou SYMBOL_SWAP_SHORT) ao transferir a posição de sábado para domingo	double
SYMBOL_MARGIN_INITIAL	Margem inicial signific	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	a a quantia de moeda de margem requerida para abrir uma posição com o volume de um lote. É usada para verificar os ativos de um cliente quando ele ou ele entra no mercado. A função SymbolInfoMarginRate() fornece informações sobre o valor da margem cobrada, dependendo do tipo e direção	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	da ordem.	
SYMBOL_MARGIN_MAINTENANCE	A margem de manutenção. Se for definida, ele representa a quantia de margem na moeda de margem do ativo cobrada de um lote. É usada parar verificar os ativos de um cliente quando o estado da conta dele/dela muda. Se a margem de manutenção é igual a 0, a margem inicial é usada.	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	A função SymbolInfoMarginRate() fornece informações sobre o valor da margem cobrada, dependendo do tipo e direção da ordem.	
SYMBOL_SESSION_VOLUME	Volume resumo de operações da sessão atual	double
SYMBOL_SESSION_TURNOVER	Rotatividade resumo da sessão atual	double
SYMBOL_SESSION_INTEREST	Juros de abertura resumo	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volume atual de ordens de Compra	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Volume atual de ordens	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de Venda	
SYMBOL_SESSION_OPEN	Preço de abertura da sessão atual	double
SYMBOL_SESSION_CLOSE	Preço de fechamento da sessão atual	double
SYMBOL_SESSION_AW	Preço médio ponderado da sessão atual	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Preço de liquidação da sessão atual	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Mínimo preço da sessão atual	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	Máximo preço da sessão atual	double
SYMBOL_MARGIN_HEDGED	Tamanho do contrato ou margem para um lote de posiçõe	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	s sobrepostas (posições com várias direções segundo um mesmo símbolo). Há duas maneiras de calcular a margem dessas posições. O método de cálculo é determinado pela corretora. Cálculo básico: • Se, para o instrumento, tiver sido estabelecida uma margem	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	inicial (SYMBOL_MARGIN_INITIAL), então, a margem de cobertura é indicada como valor absoluto (em dinheiro). • Se não se tiver estabelecido uma margem inicial (igual a 0), então, no campo SYMBOL_MARGIN_HEDGED indica-se o tama	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	nho do contrato que será usado no cálculo da margem segundo a fórmula correspondente ao tipo de instrumento de negociação (SYMBOL_TRADE_CALC_MODE). Cálculo pela maior posição: • O valor do SYMBOL_MARGIN_HEDGED não é	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	tido em conta. • Calcula-se o volume de todas as posições tanto curtas como longas segundo o instrumento. • Para cada lado calcula-se o preço médio ponderado tanto de abertura, como de conversão para a moeda do	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	depósito. • A seguir, segundo as fórmulas correspondentes ao tipo de instrumento de negociação (SYMBOL_TRADE_CALC_MODE), calcula-se a margem para o lado curto e longo. • Como valor final usa-se o maior.	
SYMBOL_PRICE_CHANGE	Alteração do preço atual	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	em relação ao final do dia anterior, expresso em porcentagem.	
SYMBOL_PRICE_VOLATILITY	Volatilidade percentual	double
SYMBOL_PRICE_THEORETICAL	Preço teórico da opção	double
SYMBOL_PRICE_DELTA	Delta da opção/garantia. Mostra quanto o preço da opção muda em unidades quando o preço do ativo subjacente mudar 1 unidade	double
SYMBOL_PRICE_THETA	Teta da opção/garantia. O número de pontos que o preço	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	da opção perderá cada dia devido a deterioração temporária, ou seja, quando a data de vencimento se aproxima	
SYMBOL_PRICE_GAMMA	Gama da opção/garantia. Mostra a taxa de alteração de delta, isto é, exibe com que rapidez ou lentidão muda o prêmio da opção	double
SYMBOL_PRICE_VEGA	Vega da opção/garantia. Mostra o número de	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	pontos que mudará o preço da opção quando a volatilidade mudar 1%	
SYMBOL_PRICE_RHO	Rô da opção/garantia. Reflete a sensibilidade do preço teórico da opção a variações de 1% da taxa de juros	double
SYMBOL_PRICE_OMEGA	Omega da opção/garantia. A elasticidade da opção é a variação percentual no preço da opção em relação à	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	variação percentual no preço do ativo subjacente	
SYMBOL_PRICE_SENSITIVITY	Sensibilidade da opção/garantia. Mostra quantos pontos deve mudar o preço do ativo subjacente da opção para que o preço da opção mude um ponto	double

Para a função SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

Identificador	Descrição	Tipo
SYMBOL_BASIS	O ativo subjacente de um derivado	string
SYMBOL_CATEGORY	Nome da categoria ou setor ao qual pertence o ativo	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
SYMBOL_COUNTRY	País para o qual pertence o instrumento financeiro	string
SYMBOL_SECTOR_NAME	Setor da economia ao qual pertence o instrumento financeiro	string
SYMBOL_INDUSTRY_NAME	Ramo da economia ou tipo de indústria ao qual pertence o instrumento financeiro	string
SYMBOL_CURRENCY_BASE	Moeda base de um ativo	string
SYMBOL_CURRENCY_PROFIT	Moeda de lucro	string
SYMBOL_CURRENCY_MARGIN	Moeda de margem	string
SYMBOL_BANK	Fornecedor da cotação atual	string
SYMBOL_DESCRIPTION	Descrição do ativo	string
SYMBOL_EXCHANGE	Nome da bolsa de valores ou da plataforma de	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	negociação em que é negociado o ativo	
SYMBOL_FORMULA	Fórmula para construir o preço de um símbolo personalizado. Se o nome do instrumento financeiro incluído na fórmula começar com um dígito ou contiver um caractere especial ("", ".", "-", "&", "#", etc.), o nome desse instrumento deverá ser colocado entre aspas. • Símbolo lo sin	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	• tético: "@ESU19"/EURCAD Spread de calendário: "Si-9.13	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	"-"Si-6.13" • índice euro: 34.388057266*pow(EURUSD,0.31	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	5 5) * p o w (E U R G B P , 0 . 3 0 5 6) * p o w (E U R J P Y , 0 . 1 8 9 1) * p o w (E U R	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	CHF,0.1113)*pow(EURSEK,0.0785)	
SYMBOL_ISIN	O nome de um ativo no sistema ISIN (International Securities Identification Number). O International Securities Identification Number é	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	um código de 12 dígitos alfanumérico que identifica de forma única um ativo financeiro. A presença desta propriedade de ativo é determinada no lado do servidor de negociação.	
SYMBOL_PAGE	Endereço de página com informações sobre o símbolo. Esse endereço será mostrado como um referência durante a visualização de propriedades do símbolo no terminal	string
SYMBOL_PATH	Caminho da árvore de ativos	string

© 2000-2024, MetaQuotes Ltd.

O gráfico de preço segundo o símbolo pode ser construído com base no preço Bid ou Last. A maneira como são geradas e exibidas as barras - no terminal - depende da escolha do preço para construir gráficos. Possíveis valores da propriedade SYMBOL_CHART_MODE são dadas na enumeração ENUM_SYMBOL_CHART_MODE

ENUM_SYMBOL_CHART_MODE

Identificador	Descrição
SYMBOL_CHART_MODE_BID	Barras são construídas segundo o preço Bid
SYMBOL_CHART_MODE_LAST	Barras são construídas segundo o preço Last

Para cada ativo vários modos de expiração de ordens pendentes podem ser especificados. Uma flag é associado a cada modo. Flags pode ser combinados usando a operação de **OR** lógico (|), por exemplo, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. A fim de verificar se um certo modo é permitido para um ativo, o resultado do**AND**(&)deve ser comparado ao flag do modo.

Se o flag SYMBOL_EXPIRATION_SPECIFIED for especificado para um ativo, então ao enviar uma ordem pendente, você pode especificar até que momento esta ordem pendente é válida.

Identificador	Valor	Descrição
SYMBOL_EXPIRATION_GTC	1	A ordem é válida durante um período de tempo ilimitado, até que ela seja explicitamente cancelada
SYMBOL_EXPIRATION_DAY	2	A ordem é válida até o fim do dia
SYMBOL_EXPIRATION_SPECIFIED	4	A hora de expiração é especificada na ordem
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	A data de expiração é especificada na ordem

Exemplo:

```
//+------------------------------------------------------------------+
//| Verifica se um modo de expiração especifico é permitido          |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- Obtém o valor da propriedade que descreve os modos de expiração permitidos
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- Retorna true, se o modo exp_type é permitido
   return((expiration&exp_type)==exp_type);
  }
```

© 2000-2024, MetaQuotes Ltd.

Se a propriedade SYMBOL_EXPIRATION_MODE tem o valor SYMBOL_EXPIRATION_GTC (ordem válida até ser cancelada), a validade das ordens pendentes e os níveis colocados StopLoss/TakeProfit são definidos adicionalmente usando a enumeração ENUM_SYMBOL_ORDER_GTC_MODE.

ENUM_SYMBOL_ORDER_GTC_MODE

Identificador	Descrição
SYMBOL_ORDERS_GTC	Ordens pendentes e níveis Stop Loss/Take Profit são validas de maneira ilimitada no tempo até a cancelação explicita
SYMBOL_ORDERS_DAILY	As ordens são válidas apenas dentro de um único dia de negociação. Após a conclusão, todos os níveis de StopLoss e TakeProfit são removidos, bem como ordens pendentes são excluídas
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	Ao mudar o dia de negociação são excluídas apenas as ordens pendentes, enquanto os níveis StopLoss e TakeProfit são salvos.

Ao enviar uma ordem, você pode especificar a política de preenchimento do volume declarado na ordem de negociação. As opções permitidas de execução de ordem por volume para cada ativo são mostradas na tabela. Para cada ativo, podem ser definidos vários modos por meio de uma combinação de flags. A combinação de flags é expressa pela operação lógica **OU** (|), por exemplo, SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Para verificar a resolução de um modo específico para um ativo, você precisa comparar o resultado de **E** (&) com o flag de modo - exemplo.

Política de preenchimento	Identificador	Valor	Descrição
Tudo/Nada	SYMBOL_FILLING_FOK	1	A ordem pode ser

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			executada apenas como volume especificado. Sen

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ãohouvervolumedoativofinanceironomercadonãof

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ors ufi cie nte nom om ent o, ao rd em nã os er á ex ec ut a

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			da. O volume necessário pode ser composto por dive

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			rsasoferta satualmente disponíveis no mercad

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			o. Comestapolítica, aoenviarumaordem, vocêdev

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			especificar o tipo de preenchimento ORDER_FILL

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			NG_FO. Apossibilidadedeusarordensfokédete

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			rminadanoserviidordenegociação.
Tudo/Parcial	SYMBOL_FILLING_IOC	2	Ordeeraceit

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			a executar a transação como volume máximo disponível

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ível nomercado, dentrodoslimitesespecificados n

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			a ordem. Se a execução completa não for possível, a

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ordem será executada como volume disponível, ou

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			olumenãoexecutadodaordemserácancelado. Co

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			mestapolítica,aoenviarumaordem,vocêdeveespe

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			cificar o tipo de preenchimento ORDER_FILLING_IO

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			C. Apossibilidade de usar ordens IOC é determina

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			dano servidor de negociação.
Passiva	SYMBOL_FILLING_BOC	4	Apolítica BoC (Boo

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			k-or-Cancel) implica que o pedido só pode ser cancel

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ocadonolivrodeofertasenãopodesserexecutado

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			imediatamente. Se a ordem pode ser executada im

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ediatamente quando colocada, ela é retirada.

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			Na verdade, esta política de execuções só pode ser

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			especificada quando o preço da ordem colocada fo

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			rpiordoqueomercadoatual.AsordensBOCsãouti

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			lizadas para negociação passiva, garantindo qu

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			e a ordem não se já a executada imediatamente após a

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			colocação, equenãoafetealiquidezatual. Ela

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			é suportada somente para ordens limite e stop-li

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			mit, ou seja, se o sinalizador SYMBOL_ORDER_MOD

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			Econtém valores SYMBOL_ORDER_LIMIT o/u SYMBO

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			L_ORDER_STOP_LIMIT.
Retornar	Sem identificador		Em caso de execução parcial

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			l, uma ordem a mercado ou limite com volume residual c

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			ontinuaemvigor,emvezdesserremovida. Comesta

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			política, a envia uma ordem, você deve especifica

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			rotipodepreenchimentoORDER_FILLING_RETURN.

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			Ordens Retornadas são permitidas no modo Market

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			Execution (execução de ordem a mercado - SYMBOL

© 2000-2024, MetaQuotes Ltd.

Política de preenchimento	Identificador	Valor	Descrição
			_TRADE_EXECUTION_MARKET).

Ao enviar a ordem de negociação usando a função OrderSend(), a devida política de execução quanto ao volume pode ser definida no campo *type_filling* na estrutura especial MqlTradeRequest, com valores válidos da enumeração ENUM_ORDER_TYPE_FILLING. Se o tipo de preenchimento não for especificado, ORDER_FILLING_RETURN será automaticamente substituído na ordem de negociação. Vale a pena ressaltar que o tipo de preenchimento ORDER_FILLING_RETURN é permitido para qualquer modo de execução, exceto para o modo "Execução a mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

Ao enviar uma ordem de negociação para ser executada **no momento atual** (time in force), é necessário ter em conta que nos mercados financeiros não existe garantia de que num dado momento para dado ativo financeiro todo o volume solicitado estará disponível ao preço pretendido. É por esse motivo que a realização de operações de negociação em tempo real é regulada pelos modos de execução com base no preço e no volume. Os modos, ou seja, as políticas de execução, definem as regras para os casos em que o preço mudou ou o volume solicitado não pode ser totalmente cumprido **no momento**.

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
Modo de execução (Request Execution)	Execução da ordem de mercado ao preço previamente recebido da corretora. Antes de uma ordem de mercado ser enviada, são solicitados à corretora os preços pelos qual ela será executada. Após serem recebidos, a execução da ordem a um determinado preço pode ser confirmada	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	ou rejeitada.	
Execução imediata (Instant Execution)	Execução imediata de uma ordem de mercado ao preço especificado. Ao enviar uma ordem de negociação para ser executada, a plataforma substitui automaticamente os preços atuais na ordem. • Se a corret	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	o r a a c e i t a r o p r e ç o , a o r d e m s e r á e x e c u t a d a . • S e a c o r r e t o r	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	a n ã o a c e i t a r o p r e ç o s o l i c i t a d o , o c o r r e a c h a m a d a " r e c o t	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	ação "(Requote) - a corretora retorna os preços aos quais	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	a ordem pode ser executada.	
Execução a mercado (Market Execution)	A decisão sobre o preço de execução é feita pela corretora sem acordo adicional com o trader. O envio de uma ordem a mercado nesta	SYMBOL_TRADE_EXECUTION_MARKET

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	modalidade implica que se aceita antecipadamente o preço a que será executada.	
Execução de bolsa (Exchange Execution)	As operações de negociação são realizadas aos preços das cotações atuais de mercado.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Antes de enviar uma ordem para ser executada no momento atual, para a correta configuração do valor ORDER_TYPE_FILLING (tipo de execução por volume), para cada ativo financeiro por meio da função SymbolInfoInteger() você pode obter o valor da propriedade SYMBOL_FILLING_MODE que mostra os tipos de preenchimento por volume (permitidos para este símbolo) como uma combinação de flags. Deve-se observar que o tipo de preenchimento ORDER_FILLING_RETURN é sempre permitido, exceto para o modo "Execução da mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

O uso de tipos de preenchimento dependendo do modo de execução pode ser representado na forma de uma tabela:

Modo de execução\Política de preenchimento	Tudo/Nada (FOK ORDER_FILLING_FOK)	Tudo/Parcial (IOC ORDER_FILLING_IOC)	Retornar (Return ORDER_FILLING_RETURN)
Execução imediata (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (independentemente da	+ (independentemente da	+ (sempre)

© 2000-2024, MetaQuotes Ltd.

Modo de execução\Política de preenchimento	Tudo/Nada (FOK ORDER_FILLING_FOK)	Tudo/Parcial (IOC ORDER_FILLING_IOC)	Retornar (Return ORDER_FILLING_RETURN)
	configuração do ativo)	configuração do ativo)	
Execuções a pedido SYMBOL_TRADE_EXECUTION_REQUEST	+ (independentemente da configuração do ativo)	+ (independentemente da configuração do ativo)	+ (sempre)
Execução a mercado SYMBOL_TRADE_EXECUTION_MARKET	+ (definido nas configurações do ativo)	+ (definido nas configurações do ativo)	- (proibido independentemente da configuração do ativo)
Execução de bolsa SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (definido nas configurações do ativo)	+ (definido nas configurações do ativo)	+ (sempre)

Para ordens pendentes, independentemente do modo de execução (SYMBOL_TRADE_EXEMODE) é necessário utilizar o tipo de preenchimento ORDER_FILLING_RETURN, uma vez que essas ordens são executadas no momento do envio. Ao usar ordens pendentes, o trader aceita com antecedência que quando as condições para realização da transação nesta ordem forem atendidas, a corretora usará o tipo de preenchimento que é suportado por dada plataforma de negociação.

Exemplo:

```
//+------------------------------------------------------------------+
//| verifica se o modo de preenchimento especificado é permitido     |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- obtemos o valor da propriedade que descreve o modo de preenchimento
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- retornamos true, se o modo fill_type estive habilitado
   return((filling&fill_type)==fill_type);
  }
```

Ao enviar um pedido de comércio usando a função OrderSend(), um tipo de ordem do enumerador ENUM_ORDER_TYPE enumeration deve ser especificada para algumas operações. Nem todos os tipos de ordens podem ser autorizadas por um símbolo específico. Propriedade SYMBOL_ORDER_MODE descreve os sinalizadores dos tipos de ordens permitidas.

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Descrição
SYMBOL_ORDER_MARKET	1	Ordens de mercado são permitidas (Buy e Sell)
SYMBOL_ORDER_LIMIT	2	Ordens de limite são permitidas (Buy Limit e Sell Limit)
SYMBOL_ORDER_STOP	4	Ordens de parada são permitidas (Buy Stop e Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Ordens de limite de parada são permitidas (Buy Stop Limit e Sell Stop Limit)
SYMBOL_ORDER_SL	16	Stop Loss é permitido
SYMBOL_ORDER_TP	32	Take Profit é permitido
SYMBOL_ORDER_CLOSEBY	64	Autorização para fechar a posição usando outra oposta, quer dizer, usando uma posição aberta na direção oposta, no mesmo mesmo instrumento. A propriedade é definida para contas com sistema de cobertura (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

Exemplo:

```
//+------------------------------------------------------------------+
//| A função imprime tipos de ordens permitidas para um símbolo      |
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- receber o valor da propriedade descrevendo os tipos de ordens permitidas
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- verificar se há ordens de mercado (Execução de Mercado)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Ordens de mercado são permitidas (Buy e Sell)");
//--- verificar se há ordens de Limite
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Ordens Buy Limit e Sell Limit são permitidas");
//--- verificar se há ordens de Parada
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Ordens Buy Stop e Sell Stop são permitidas");
//--- verificar se há ordens Stop Limit
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Ordens Buy Stop Limit e Sell Stop Limit são permitidas");
//--- verificar se a colocação de uma ordem Stop Loss é permitida
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": Ordens de Stop Loss são permitidas");
```

© 2000-2024, MetaQuotes Ltd.

```
//--- verificar se a colocação de uma ordem Take Profit é permitida
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Ordens de Take Profit são permitidas");
//--- verificar se a permissão para fechar uma posição usando uma posição oposta
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": Permitido fechamento com posição oposta");
//---
  }
```

A enumeração ENUM_SYMBOL_CALC_MODE é usada para obter informação sobre como os requerimentos de margem para um ativo são calculados.

ENUM_SYMBOL_CALC_MODE

Identificador	Descrição	Fórmula
SYMBOL_CALC_MODE_FOREX	Modo Forex - cálculo de lucro e margem para Forex	Margin: Lots * Contract_Size / Leverage * Margin_Rate Profit: (close_price - open_price) * Contract_Size*Lots
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	Forex No Leverage mode - cálculo de lucro e margem para o mercado Forex sem levar	Margin: Lots * Contract_Size * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	em conta a alavancagem	
SYMBOL_CALC_MODE_FUTURES	Modo Futuros - cálculo de margem e lucro para futuros	Margin: Lots * InitialMargin * Margin_Rate Profit: (close_price - open_price) * TickPrice / TickSize*Lots
SYMBOL_CALC_MODE_CFD	Modo CFD - cálculo de margem e lucro para CFD	Margin: Lots * ContractSize * MarketPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDINDEX	Modo índice CFD - cálculo de margem e lucro para CFD por índices	Margin: (Lots * ContractSize * MarketPrice) * TickPrice / TickSize * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	Modo Alavancado	Margin: (Lots * ContractSize * MarketPrice) / Leverage * Margin_Rate

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	CFD - cálculo de margem e lucro para CFD em negociação alavancada	Profit: (close_price-open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_STOCKS	Modo Exchange - cálculo de margem e lucro para negociação de títulos na bolsa de valores	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	Modo Futures - cálculo da margem e lucro para os contratos	Margin: Lots * InitialMargin * Margin_Rate ou Lots * MaintenanceMargin * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	futuros de negociação na bolsa de valores	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	Modo FORTS Futures - cálculo de margem e lucro para futuros contratos de comercialização nos FORTS. A margem pode ser reduzida pela quantidade de desvio MarginDiscount de	Margin: Lots * InitialMargin * Margin_Rate ou Lots * MaintenanceMargin * Margin_Rate * Margin_Rate Profit: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	acordo com as regras seguintes: 1. Se o preço de uma posição long (ordem BUY) é menor do que o preço estimado, MarginDiscount = Lots*((PriceSettle-PriceOrder)*TickPrice/TickSize) 2. Se o preç	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	o de uma posição short (ordem SELL) é superior ao preço estimado, MarginDiscount = Lots*((PriceOrder-PriceSettle)*TickPrice/TickSize) onde: ○ PriceSettle-p	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	○ Preço estimado (limpando) das essas ãoa nterior;	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	r - preço médio ponderado de posição ou preço de abertura d	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	efinidonaordem(pedido); ○ TickPrice - Preço de tick (cus	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	todamudança do preço por um ponto) ○ TickSize - tamanho otic	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	k (passo de alteração de preço mínima)	
SYMBOL_CALC_MODE_EXCH_BONDS	Exchange Bonds mode - cálculo da garantia e do lucro para negociar título	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	s na bolsa de valores	
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Exchange MOEX Stocks mode - cálculo da garantia e do lucro para negociar títulos na Bolsa de Moscou	Margin: Lots * ContractSize * LastPrice * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Exchange MOEX Bonds mode - cálculo da garantia e do lucro para negociar títulos de	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	dívida na Bolsa de Moscou	
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a trading account. The market value of an open position is calculated based on the volume, current market price, contract size	Margin: no Profit: no Market Value: Lots * ContractSize * MarketPrice * LiquidityRate

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	and liquidity ratio. The value is included into Assets, which are added to Equity. Open positions of such symbols increase the Free Margin amount and are used as additional margin (collateral) for open positions of	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Fórmula
	tradable instruments.	

Existem vários modos de negociação de ativos. Informação sobre modos de negociação de um certo ativo é refletido nos valores da enumeração ENUM_SYMBOL_TRADE_MODE.

ENUM_SYMBOL_TRADE_MODE

Identificador	Descrição
SYMBOL_TRADE_MODE_DISABLED	Negociação está desabilitada para o ativo
SYMBOL_TRADE_MODE_LONGONLY	Permitido somente posições compradas
SYMBOL_TRADE_MODE_SHORTONLY	Permitido somente posições vendidas
SYMBOL_TRADE_MODE_CLOSEONLY	Permitido somente operações de fechamento de posição
SYMBOL_TRADE_MODE_FULL	Sem restrições de negociação

Os possíveis modos de execução de uma operação (deal) para um certo ativo são definidos na enumeração ENUM_SYMBOL_TRADE_EXECUTION.

ENUM_SYMBOL_TRADE_EXECUTION

Identificador	Descrição
SYMBOL_TRADE_EXECUTION_REQUEST	Execução por solicitação
SYMBOL_TRADE_EXECUTION_INSTANT	Execução instantânea
SYMBOL_TRADE_EXECUTION_MARKET	Execução a Mercado
SYMBOL_TRADE_EXECUTION_EXCHANGE	Execução em um sistema de negociação externo (Exchange)

Métodos de cálculo de swap na transferência de posição são especificados na enumeração ENUM_SYMBOL_SWAP_MODE. O método de cálculo de swap determina a unidade de medida dos parâmetros SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT. Por exemplo, se swaps são cobrados na moeda de depósito do cliente, então os valores destes parâmetros são especificados como uma quantia de dinheiro na moeda de depósito do cliente.

ENUM_SYMBOL_SWAP_MODE

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
SYMBOL_SWAP_MODE_DISABLED	Desabilita swaps (sem swaps)
SYMBOL_SWAP_MODE_POINTS	Swaps são cobrados em pontos
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swaps são cobrados em dinheiro na moeda base do ativo
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swaps são cobrados em dinheiro na moeda de margem do ativo
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swaps são cobrados em dinheiro na moeda de depósito do cliente
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swaps são cobrados como juro anual especificado a partir do preço de instrumento no cálculo do swap (ano bancário padrão é 360 dias)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swaps são cobradas como juro anual especificado a partir do preço de abertura da posição (ano bancário padrão é 360 dias)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swaps são cobradas através de posições reabertas Ao final de um dia de negociação a posição é encerrada. No dia seguinte ela é reaberta através do preço de fechamento +/- um número específico de pontos (parâmetros SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	Swaps são cobradas através de posições reabertas Ao final de um dia de negociação a posição é encerrada. No dia seguinte ela é reaberta através do preço de Venda (Bid) atual +/- um número específico de pontos (parâmetros SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)

Os valores da enumeração ENUM_DAY_OF_WEEK são usados para especificar dias da semana.

ENUM_DAY_OF_WEEK

Identificador	Descrição
SUNDAY	Domingo
MONDAY	Segunda-feira
TUESDAY	Terça-feira
WEDNESDAY	Quarta-feira
THURSDAY	Quinta-feira
FRIDAY	Sexta-feira

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
SATURDAY	Sábado

An option is a contract, which gives the right, but not the obligation, to buy or sell an underlying asset (goods, stocks, futures, etc.) at a specified price on or before a specific date. The following enumerations describe option properties, including the option type and the right arising from it.

ENUM_SYMBOL_OPTION_RIGHT

Identifier	Descrição
SYMBOL_OPTION_RIGHT_CALL	A call option gives you the right to buy an asset at a specified price
SYMBOL_OPTION_RIGHT_PUT	A put option gives you the right to sell an asset at a specified price

ENUM_SYMBOL_OPTION_MODE

Identifier	Descrição
SYMBOL_OPTION_MODE_EUROPEAN	A opção europeia só pode ser exercida em uma data especificada (validade, data de assinatura, data de entrega)
SYMBOL_OPTION_MODE_AMERICAN	A opção americana pode ser exercida em qualquer dia de negociação ou antes do termo. O prazo dentro do qual o comprador pode exercer a opção é especificado para ele

Os instrumentos financeiros são distribuídos por setores da economia. Um setor de uma economia é definido por suas características, objetivos econômicos, funções e comportamento gerais, o que o permite diferenciar de outras parcelas da economia. ENUM_SYMBOL_SECTOR lista os setores da economia aos quais um instrumento comercial pode pertencer.

ENUM_SYMBOL_SECTOR

Identificador	Descrição
SECTOR_UNDEFINED	Indefinido
SECTOR_BASIC_MATERIALS	Commodities
SECTOR_COMMUNICATION_SERVICES	Serviços de comunicação
SECTOR_CONSUMER_CYCLICAL	Consumo cíclico de demanda
SECTOR_CONSUMER_DEFENSIVE	Consumo básico
SECTOR_CURRENCY	Moedas

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
SECTOR_CURRENCY_CRYPTO	Criptomoedas
SECTOR_ENERGY	Energia
SECTOR_FINANCIAL	Finanças
SECTOR_HEALTHCARE	Cuidados de saúde
SECTOR_INDUSTRIALS	Indústria
SECTOR_REAL_ESTATE	Imobiliário
SECTOR_TECHNOLOGY	Tecnologia
SECTOR_UTILITIES	Serviços públicos
SECTOR_INDEXES	Índices
SECTOR_COMMODITIES	Mercadorias

Cada instrumento financeiro pode pertencer a um tipo específico de indústria ou setor. Um setor da economia é um conjunto de empresas que produzem produtos homogêneos ou específicos usando o mesmo tipo de tecnologia. ENUM_SYMBOL_INDUSTRY lista os tipos de indústria aos quais um instrumento comercial pode pertencer.

ENUM_SYMBOL_INDUSTRY

Identificador	Descrição
INDUSTRY_UNDEFINED	Indefinido
Commodities	
INDUSTRY_AGRICULTURAL_INPUTS	Recursos agrícolas
INDUSTRY_ALUMINIUM	Alumínio
INDUSTRY_BUILDING_MATERIALS	Materiais de construção
INDUSTRY_CHEMICALS	Produtos químicos
INDUSTRY_COKING_COAL	Coque
INDUSTRY_COPPER	Cobre
INDUSTRY_GOLD	Ouro
INDUSTRY_LUMBER_WOOD	Produção de madeira
INDUSTRY_INDUSTRIAL_METALS	Outros metais e mineração industrial
INDUSTRY_PRECIOUS_METALS	Outros metais preciosos e mineração
INDUSTRY_PAPER	Produtos de papel e celulose

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_SILVER	Prata
INDUSTRY_SPECIALTY_CHEMICALS	Produtos químicos especializados
INDUSTRY_STEEL	Aço
Serviços de comunicação	
INDUSTRY_ADVERTISING	Agencias de propaganda
INDUSTRY_BROADCASTING	Transmissão
INDUSTRY_GAMING_MULTIMEDIA	Jogos eletrônicos e multimídia
INDUSTRY_ENTERTAINMENT	Entretenimento
INDUSTRY_INTERNET_CONTENT	Conteúdo e informações da Internet
INDUSTRY_PUBLISHING	Editorial
INDUSTRY_TELECOM	Serviços de telecomunicações
Consumo cíclico de demanda	
INDUSTRY_APPAREL_MANUFACTURING	Fabricação de vestuário
INDUSTRY_APPAREL_RETAIL	Varejo de vestuário
INDUSTRY_AUTO_MANUFACTURERS	Fabricantes de automóveis
INDUSTRY_AUTO_PARTS	Autopeças
INDUSTRY_AUTO_DEALERSHIP	Concessionárias de carros e caminhões
INDUSTRY_DEPARTMENT_STORES	Lojas de departamento
INDUSTRY_FOOTWEAR_ACCESSORIES	Calçados e acessórios
INDUSTRY_FURNISHINGS	Móveis, acessórios e eletrodomésticos
INDUSTRY_GAMBLING	Empresas de jogos de azar
INDUSTRY_HOME_IMPROV_RETAIL	Comércio varejista de artigos para o lar
INDUSTRY_INTERNET_RETAIL	Varejo on-line
INDUSTRY_LEISURE	Lazer
INDUSTRY_LODGING	Alojamento
INDUSTRY_LUXURY_GOODS	Produtos de luxo
INDUSTRY_PACKAGING_CONTAINERS	Embalagem
INDUSTRY_PERSONAL_SERVICES	Serviços pessoais
INDUSTRY_RECREATIONAL_VEHICLES	Veículos de lazer
INDUSTRY_RESIDENT_CONSTRUCTION	Construção de moradias

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_RESORTS_CASINOS	Resorts e cassinos
INDUSTRY_RESTAURANTS	Restaurantes
INDUSTRY_SPECIALTY_RETAIL	Varejo especializado
INDUSTRY_TEXTILE_MANUFACTURING	Produção têxtil
INDUSTRY_TRAVEL_SERVICES	Serviços turísticos
Consumo básico	
INDUSTRY_BEVERAGES_BREWERS	Bebidas - Cervejeiros
INDUSTRY_BEVERAGES_NON_ALCO	Bebidas - Não alcoólicas
INDUSTRY_BEVERAGES_WINERIES	Bebidas - Vinícolas e Destilarias
INDUSTRY_CONFECTIONERS	Confeiteiros
INDUSTRY_DISCOUNT_STORES	Lojas de desconto
INDUSTRY_EDUCATION_TRAINIG	Educação e formação
INDUSTRY_FARM_PRODUCTS	Produtos agrícolas
INDUSTRY_FOOD_DISTRIBUTION	Distribuição de alimentos
INDUSTRY_GROCERY_STORES	Mercearias e supermercados
INDUSTRY_HOUSEHOLD_PRODUCTS	Produtos domésticos e pessoais
INDUSTRY_PACKAGED_FOODS	Produtos embalados
INDUSTRY_TOBACCO	Tabaco
Energia	
INDUSTRY_OIL_GAS_DRILLING	Perfuração de poços de petróleo e gás
INDUSTRY_OIL_GAS_EP	Extração e processamento de petróleo e gás
INDUSTRY_OIL_GAS_EQUIPMENT	Equipamento e serviços de petróleo e gás
INDUSTRY_OIL_GAS_INTEGRATED	Empresas integradas de petróleo e gás
INDUSTRY_OIL_GAS_MIDSTREAM	Transporte de petróleo e gás
INDUSTRY_OIL_GAS_REFINING	Refinação e comercialização de petróleo e gás
INDUSTRY_THERMAL_COAL	Carvão térmico
INDUSTRY_URANIUM	Urano
Finanças	
INDUSTRY_EXCHANGE_TRADED_FUND	Fundo de câmbio
INDUSTRY_ASSETS_MANAGEMENT	Gestão de ativos

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_BANKS_DIVERSIFIED	Bancos - Diversificados
INDUSTRY_BANKS_REGIONAL	Bancos - Regionais
INDUSTRY_CAPITAL_MARKETS	Mercados financeiros
INDUSTRY_CLOSE_END_FUND_DEBT	Fundo fechado - Instrumentos de dívida
INDUSTRY_CLOSE_END_FUND_EQUITY	Fundo fechado - Ações
INDUSTRY_CLOSE_END_FUND_FOREIGN	Fundo fechado - Estrangeiros
INDUSTRY_CREDIT_SERVICES	Serviços de crédito
INDUSTRY_FINANCIAL_CONGLOMERATE	Conglomerados financeiros
INDUSTRY_FINANCIAL_DATA_EXCHANGE	Dados financeiros e bolsa
INDUSTRY_INSURANCE_BROKERS	Corretores de seguros
INDUSTRY_INSURANCE_DIVERSIFIED	Seguros - Diversificados
INDUSTRY_INSURANCE_LIFE	Seguros - Vida
INDUSTRY_INSURANCE_PROPERTY	Seguros - Imobiliário e acidentes
INDUSTRY_INSURANCE_REINSURANCE	Seguros - Resseguros
INDUSTRY_INSURANCE_SPECIALTY	Seguros - Especial
INDUSTRY_MORTGAGE_FINANCE	Financiamento hipotecário
INDUSTRY_SHELL_COMPANIES	Empresas Shell
Cuidados de saúde	
INDUSTRY_BIOTECHNOLOGY	Biotecnologia
INDUSTRY_DIAGNOSTICS_RESEARCH	Diagnóstico e pesquisa
INDUSTRY_DRUGS_MANUFACTURERS	Fabricação farmacêutica - Geral
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	Fabricação farmacêutica - Especialidades e genéricos
INDUSTRY_HEALTHCARE_PLANS	Planos de saúde
INDUSTRY_HEALTH_INFORMATION	Serviços de informação em saúde
INDUSTRY_MEDICAL_FACILITIES	Instituições médicas
INDUSTRY_MEDICAL_DEVICES	Equipamento médico
INDUSTRY_MEDICAL_DISTRIBUTION	Distribuidores médicos
INDUSTRY_MEDICAL_INSTRUMENTS	Instrumentos e suprimentos médicos
INDUSTRY_PHARM_RETAILERS	Varejistas farmacêuticos
Indústria	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_AEROSPACE_DEFENSE	Aeroespacial e defesa
INDUSTRY_AIRLINES	Companhias aéreas
INDUSTRY_AIRPORTS_SERVICES	Aeroportos e transporte aéreo
INDUSTRY_BUILDING_PRODUCTS	Materiais e equipamentos de construção
INDUSTRY_BUSINESS_EQUIPMENT	Equipamentos e materiais comerciais
INDUSTRY_CONGLOMERATES	Conglomerados
INDUSTRY_CONSULTING_SERVICES	Serviços de consultoria
INDUSTRY_ELECTRICAL_EQUIPMENT	Equipamento elétrico e peças de reposição
INDUSTRY_ENGINEERING_CONSTRUCTION	Engenharia e construção
INDUSTRY_FARM_HEAVY_MACHINERY	Equipamento de construção agrícola e pesada
INDUSTRY_INDUSTRIAL_DISTRIBUTION	Distribuidores industriais
INDUSTRY_INFRASTRUCTURE_OPERATIONS	Operações de infraestrutura
INDUSTRY_FREIGHT_LOGISTICS	Transporte e logística integrados
INDUSTRY_MARINE_SHIPPING	Remessa
INDUSTRY_METAL_FABRICATION	Produção de metal
INDUSTRY_POLLUTION_CONTROL	Controle de contaminação e limpeza
INDUSTRY_RAILROADS	Ferrovias
INDUSTRY_RENTAL_LEASING	Aluguel e locação
INDUSTRY_SECURITY_PROTECTION	Segurança e proteção
INDUSTRY_SPEALITY_BUSINESS_SERVICES	Serviços empresariais especializados
INDUSTRY_SPEALITY_MACHINERY	Máquinas industriais especializadas
INDUSTRY_STUFFING_EMPLOYMENT	Serviços de emprego
INDUSTRY_TOOLS_ACCESSORIES	Ferramentas e inventário
INDUSTRY_TRUCKING	Transporte de carga
INDUSTRY_WASTE_MANAGEMENT	Gestão de resíduos
Imobiliário	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	Imóveis - Construção
INDUSTRY_REAL_ESTATE_DIVERSIFIED	Imóveis - Diversificado
INDUSTRY_REAL_ESTATE_SERVICES	Serviços imobiliários
INDUSTRY_REIT_DIVERSIFIED	Fundo de investimento imobiliário - Diversificado

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_REIT_HEALTCARE	Fundo de investimento imobiliário - Instituições Médicas
INDUSTRY_REIT_HOTEL_MOTEL	Fundo de investimento imobiliário - Hotéis
INDUSTRY_REIT_INDUSTRIAL	Fundo de investimento imobiliário - Indústria
INDUSTRY_REIT_MORTAGE	Fundo de investimento imobiliário - Hipoteca
INDUSTRY_REIT_OFFICE	Fundo de investimento imobiliário - Escritórios
INDUSTRY_REIT_RESIDENTAL	Fundo de investimento imobiliário - Residencial
INDUSTRY_REIT_RETAIL	Fundo de investimento imobiliário - Varejo
INDUSTRY_REIT_SPECIALITY	Fundo de investimento imobiliário - Instalações especiais
Tecnologia	
INDUSTRY_COMMUNICATION_EQUIPMENT	Equipamento de comunicação
INDUSTRY_COMPUTER_HARDWARE	Equipamento informático
INDUSTRY_CONSUMER_ELECTRONICS	Eletrônicos de consumo
INDUSTRY_ELECTRONIC_COMPONENTS	Componentes eletrônicos
INDUSTRY_ELECTRONIC_DISTRIBUTION	Distribuição de eletrônicos e computadores
INDUSTRY_IT_SERVICES	Serviços de tecnologia da informação
INDUSTRY_SCIENTIFIC_INSTRUMENTS	Ferramentas científicas e técnicas
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	Equipamentos e materiais semicondutores
INDUSTRY_SEMICONDUCTORS	Semicondutores
INDUSTRY_SOFTWARE_APPLICATION	Software - Aplicativos
INDUSTRY_SOFTWARE_INFRASTRUCTURE	Software - Infraestrutura
INDUSTRY_SOLAR	Energia solar
Serviços públicos	
INDUSTRY_UTILITIES_DIVERSIFIED	Serviços públicos - Diversificado
INDUSTRY_UTILITIES_POWERPRODUCERS	Serviços públicos - Produtores de energia independentes
INDUSTRY_UTILITIES_RENEWABLE	Serviços públicos - Energia renovável
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	Serviços públicos - Companhias elétricas reguladas
INDUSTRY_UTILITIES_REGULATED_GAS	Serviços públicos - Empresas de gás reguladas

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
INDUSTRY_UTILITIES_REGULATED_WATER	Serviços públicos - Companhias elétricas reguladas

© 2000-2024, MetaQuotes Ltd.

Propriedades da Conta

Para obter informações sobre a conta atual existem várias funções: AccountInfoInteger(), AccountInfoDouble() e AccountInfoString(). Os valores dos parâmetros destas funções podem aceitar valores das enumerações de ENUM_ACCOUNT_INFO.

Para a função AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

Identificador	Descrição	Tipo
ACCOUNT_LOGIN	Número da conta	long
ACCOUNT_TRADE_MODE	Modo de negociação da conta	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Alavancagem da conta	long
ACCOUNT_LIMIT_ORDERS	Número permitido máximo de ordens pendentes ativas	int
ACCOUNT_MARGIN_SO_MODE	Modo para definir a margem permitida mínima	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	Negociação permitida para a conta atual	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
ACCOUNT_TRADE_EXPERT	Negociação permitida para um Expert Advisor	bool
ACCOUNT_MARGIN_MODE	Modo de cálculo da margem	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	Número de casas decimais, para a moeda da conta, necessário para exibir com precisão os resultados da negociação	int
ACCOUNT_FIFO_CLOSE	Indica que a posição pode ser fechada apenas pelo princípio FIFO. Se o valor da proprie	bool

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	dade for igual a true, será permitido fechar a posição de cada símbolo apenas segundo a ordem em que foram abertas, quer dizer, primeiro, a mais antiga, em seguida, a mais recente e assim por diante. Ao tentar fechar posições em uma ordem diferente, aparecerá um erro. Para contas	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	sem registo de posições usando cobertura (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), a propriedade sempre será igual a false	
ACCOUNT_HEDGE_ALLOWED	Indica que são permitidas posições opostas de um símbolo de cada vez	bool

Para a função AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

Identificador	Descrição	Tipo
ACCOUNT_BALANCE	Saldo da conta na	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	moeda de depósito	
ACCOUNT_CREDIT	Crédito da conta na moeda de depósito	double
ACCOUNT_PROFIT	Lucro atual de uma conta na moeda de depósito	double
ACCOUNT_EQUITY	Saldo a mercado da conta na moeda de depósito	double
ACCOUNT_MARGIN	Margem da conta usada na moeda de depósito	double
ACCOUNT_MARGIN_FREE	Margem livre de uma conta na moeda	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	de depósito	
ACCOUNT_MARGIN_LEVEL	Nível de margem de conta em percentagem	double
ACCOUNT_MARGIN_SO_CALL	Nível de chamada de margem. Dependendo da definição, ACCOUNT_MARGIN_SO_MODE é expresso em percentagem ou em moeda de depósito	double
ACCOUNT_MARGIN_SO_SO	Nível de margem de Stop Out (encerramento forçado). Depend	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	endo da definição, ACCOUNT_MARGIN_SO_MODE é expresso em percentagem ou em moeda de depósito	
ACCOUNT_MARGIN_INITIAL	Margem inicial. O montante reservado numa conta para cobrir a margem de todas as ordens pendentes	double
ACCOUNT_MARGIN_MAINTENANCE	Margem de manutenção. O patrimônio líquido mínimo reservado numa	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	conta para cobrir o valor mínimo de todas as posições em aberto	
ACCOUNT_ASSETS	Os ativos atuais de uma conta	double
ACCOUNT_LIABILITIES	As responsabilidades atuais de uma conta	double
ACCOUNT_COMMISSION_BLOCKED	O valor da comissão bloqueada atual numa conta	double

Para a função AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

Identificador	Descrição	Tipo
ACCOUNT_NAME	Nome do cliente	string
ACCOUNT_SERVER	Nome do servido	string

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	r de negociação	
ACCOUNT_CURRENCY	Moeda da conta	string
ACCOUNT_COMPANY	Nome de uma empresa que serve a conta	string

Existem vários tipos de contas que podem ser abertos em um servidor de negociação. O tipo de conta sobre o qual um programa MQL5 está sendo executado pode ser encontrado usando a enumeração ENUM_ACCOUNT_TRADE_MODE.

ENUM_ACCOUNT_TRADE_MODE

Identificador	Descrição
ACCOUNT_TRADE_MODE_DEMO	Conta de Demonstração
ACCOUNT_TRADE_MODE_CONTEST	Conta de torneio
ACCOUNT_TRADE_MODE_REAL	Conta real

No caso do saldo a mercado não ser suficiente para manter posições em aberto ocorre a situação de Stop Out, ou seja o encerramento forçado. O nível de margem mínimo no qual o Stop Out ocorre pode ser definido em percentagem ou em termos monetários. Para descobrir a configuração de modo da conta use a enumeração ENUM_ACCOUNT_STOPOUT_MODE.

ENUM_ACCOUNT_STOPOUT_MODE

Identificador	Descrição
ACCOUNT_STOPOUT_MODE_PERCENT	Modo stop out da conta em percentagem
ACCOUNT_STOPOUT_MODE_MONEY	Modo stop out da conta em dinheiro

ENUM_ACCOUNT_MARGIN_MODE

Identificador	Descrição
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Usa-se para o mercado de balcão ao registrar as posições no modo "compensação" (segundo um símbolo pode existir apenas uma posição). O cálculo da margem é realizado com base no tipo de instrumento (SYMBOL_TRADE_CALC_MODE).

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
ACCOUNT_MARGIN_MODE_EXCHANGE	Usa-se para o mercado de bolsa. O cálculo da margem é realizado com base nos descontos indicados nas configurações dos instrumentos. Os descontos são estabelecidos pela corretora, no entanto não podem ser inferiores aos valores determinados pela bolsa.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Usa-se para o mercado de balcão ao ser realizado o registro independente de posições ("cobertura", segundo um símbolo, podem existir várias posições). O cálculo da margem realiza-se com base no tipo de instrumento (SYMBOL_TRADE_CALC_MODE) e tendo em conta o tamanho da margem de cobertura (SYMBOL_MARGIN_HEDGED).

Os dados do tipo float são exibidos com 5 dígitos após o ponto decimal.

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Nome da empresa
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- Nome do cliente
   string name=AccountInfoString(ACCOUNT_NAME);
//--- Número da conta
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- Nome do servidor
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- Moeda da conta
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- Conta demo, de torneio ou real
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- Agora transforma o valor da enumeração em uma forma inteligível
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
         trade_mode="concurso";
         break;
      default:
         trade_mode="real";
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- Stop Out é definida em percentagem ou dinheiro
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- Obtém os valores dos níveis quando a Chamada de Margem e Stop Out (encerramento
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- Exibe informações resumidas sobre a conta
   PrintFormat("A conta do do cliente '%s' #%d %s aberta em '%s' no servidor '%s'",
               name,login,trade_mode,company,server);
   PrintFormat("Moeda da conta - %s, níveis de MarginCall e StopOut são configurados e
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"porcentagem":"
   PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2024, MetaQuotes Ltd.

Estatísticas de Teste

Depois que o teste é finalizado, diferentes parâmetros das estatísticas dos resultados das negociações são calculados. Os valores dos parâmetros podem ser obtidos usando a função TesterStatistics(), através da especificação do parâmetro ID a partir da enumeração ENUM_STATISTICS.

Embora dois tipo de parâmetros (int e double) serem usados nos cálculos das estáticas, a função retorna todos os valores na forma double. Todos os valores estatísticos do tipo double são expressos na moeda corrente por default, a mesmo que especificado de outro modo.

ENUM_STATISTICS

ID	Descrição do parâmetro estatístico	Tipo
STAT_INITIAL_DEPOSIT	O valor do depósito inicial	double
STAT_WITHDRAWAL	Dinheiro sacado da conta	double
STAT_PROFIT	Lucro líquido após o teste, a soma de STAT_GROSS_PROFIT e STAT_GROSS_LOSS (STAT_GROSS_LOSS é sempre menor ou igual a zero)	double
STAT_GROSS_PROFIT	Lucro total, a soma de todas as negociações lucrativas (positivas). O valor é maior ou igual a zero	double
STAT_GROSS_LOSS	Prejuízo total, a soma de todas as negociações negativas. O valor é menor ou igual a zero	double
STAT_MAX_PROFITTRADE	Lucro máximo - o maior valor dentre todas as negociações lucrativas O valor é maior ou igual a zero	double
STAT_MAX_LOSSTRADE	Prejuízo Máximo - o menor valor dentre todos as negociações (trades) perdedoras O valor é menor ou igual a zero	double

© 2000-2024, MetaQuotes Ltd.

ID	Descrição do parâmetro estatístico	Tipo
STAT_CONPROFITMAX	Lucro máximo em uma série de negociações lucrativas. O valor é maior ou igual a zero	double
STAT_CONPROFITMAX_TRADES	O número de negociações que formaram oSTAT_CONPROFITMAX(lucro máximo em uma série de negociações lucrativas)	int
STAT_MAX_CONWINS	O lucro total da mais longa série de negociações lucrativas	double
STAT_MAX_CONPROFIT_TRADES	O número de negociações na mais longa série de negociações lucrativasSTAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Prejuízo máximo em uma série de negociações perdedoras. O valor é menor ou igual a zero	double
STAT_CONLOSSMAX_TRADES	O número de negociações (trades) que formaram oSTAT_CONLOSSMAX(máximo prejuízo em uma série de negociações perdedoras)	int
STAT_MAX_CONLOSSES	O total de prejuízo da mais longa série de negociações perdedoras	double
STAT_MAX_CONLOSS_TRADES	O número de negociações na mais longa série de negociações perdedoras	int

© 2000-2024, MetaQuotes Ltd.

ID	Descrição do parâmetro estatístico	Tipo
	STAT_MAX_CONLOSSES	
STAT_BALANCEMIN	Valor do saldo mínimo	double
STAT_BALANCE_DD	Máxima queda de saldo em termos monetários Em um processo de negociação, um saldo pode sofrer numerosas quedas; aqui o maior valor é tomado	double
STAT_BALANCEDD_PERCENT	Queda de saldo como uma percentagem que foi registrada no momento da máxima queda do saldo em termos monetários (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	Máxima queda de saldo com uma percentagem. Em um processo de negociação, um saldo pode sofrer numerosas quedas, para cada qual a valor da queda relativa em percentagem é calculada. O maior valor é retornado	double
STAT_BALANCE_DD_RELATIVE	Queda de saldo em termos monetários que foi registrada no momento da máxima queda do saldo como uma percentagem (STAT_BALANCE_DDREL_PERCENT).	double
STAT_EQUITYMIN	Mínimo valor do saldo a mercado	double
STAT_EQUITY_DD	Máxima queda do saldo a mercado em termos monetários. Em um processo de	double

© 2000-2024, MetaQuotes Ltd.

ID	Descrição do parâmetro estatístico	Tipo
	negociação, numerosas quedas podem ocorrer no saldo a mercado; aqui o maior valor é tomado	
STAT_EQUITYDD_PERCENT	Queda em percentagem que foi registrada no momento da máxima queda do saldo a mercado em termos monetários (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	Máxima queda do saldo a mercado como uma percentagem. Em um processo de negociação, um saldo a mercado pode sofrer numerosas quedas, para cada qual o valor da queda relativa em percentagem é calculado. O maior valor é retornado	double
STAT_EQUITY_DD_RELATIVE	Queda do saldo a mercado em termos monetários que foi registrada no momento da máxima queda do saldo a mercado em percentagem (STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	Recompensa esperada	double
STAT_PROFIT_FACTOR	Fator de lucro, igual a razão de STAT_GROSS_PROFIT/STAT_GROSS_LOSS. Se STAT_GROSS_LOSS=0, o fator de lucro é igual aDBL_MAX	double

© 2000-2024, MetaQuotes Ltd.

ID	Descrição do parâmetro estatístico	Tipo
STAT_RECOVERY_FACTOR	Fator de recuperação, igual a razão de STAT_PROFIT/STAT_BALANCE_DD	double
STAT_SHARPE_RATIO	Razão Sharpe	double
STAT_MIN_MARGINLEVEL	Valor mínimo do nível de margem	double
STAT_CUSTOM_ONTESTER	O valor do critério de otimização customizado calculado retornado através da função OnTester()	double
STAT_DEALS	Número de operações (deals)	int
STAT_TRADES	Número de negociações (trades)	int
STAT_PROFIT_TRADES	Negociações compradas lucrativas	int
STAT_LOSS_TRADES	Negociações perdedoras	int
STAT_SHORT_TRADES	Negociações Vendidas	int
STAT_LONG_TRADES	Negociações Compradas	int
STAT_PROFIT_SHORTTRADES	Negociações vendidas lucrativas	int
STAT_PROFIT_LONGTRADES	Negociações compradas lucrativas	int
STAT_PROFITTRADES_AVGCON	Comprimento médio de uma série de negociações lucrativas	int
STAT_LOSSTRADES_AVGCON	Comprimento médio de uma série de negociações perdedores	int
STAT_COMPLEX_CRITERION	Critério de otimização complexo	

© 2000-2024, MetaQuotes Ltd.

Constantes de Negociação (Trade)

As várias constantes usadas para programar estratégias de negociação estão divididas nos seguintes grupos:

- Propriedades de banco de dados de histórico – recebimento de informações gerais sobre um ativo;
- Propriedades de ordem – obtenção de informações sobre ordens de negociação;
- Propriedades de posição – obtenção de informações sobre posições correntes;
- Propriedades de uma operação (deal) – obtenção de informações sobre operações (deals);
- Tipos de operações de negociação (trade) – descrição das operações de negociação disponíveis;
- Tipos de transação de negociação - descrição dos possíveis tipos de transações de negociação;
- Ordens de negociação no DOM (Profundidade de Mercado) – separação de ordens de acordo com a direção de uma operação solicitada.

© 2000-2024, MetaQuotes Ltd.

Propriedades de uma Base Histórica de Dados

Ao acessar uma série de tempo a função SeriesInfoInteger() é usada para obter informações adicionais de um ativo (symbol). Um identificador de uma propriedade solicitada é passada como parâmetro da função. O identificar pode ser um dos valores do ENUM_SERIES_INFO_INTEGER.

ENUM_SERIES_INFO_INTEGER

Identificador	Descrição	Tipo
SERIES_BARS_COUNT	Quantidade de barras para o ativo-período para o momento corrente	long
SERIES_FIRSTDATE	A primeira data para o ativo-período para o momento corrente	datetime
SERIES_LASTBAR_DATE	Hora de abertura da última barra do ativo-período	datetime
SERIES_SERVER_FIRSTDATE	A primeira data no histórico do ativo no servidor independentemente da janela de tempo	datetime
SERIES_TERMINAL_FIRSTDATE	A primeira data no histórico do ativo no terminal cliente, independentemente da janela de tempo	datetime
SERIES_SYNCHRONIZED	Flag de sincronização de dados de um ativo/período para o momento corrente	bool

© 2000-2024, MetaQuotes Ltd.

Propriedades de uma Ordem

As solicitações para executar operações de negociação são formalizadas como ordens. Cada ordem tem uma variedade de propriedades para leitura. As informações sobre elas podem ser obtidas usando as funções OrderGet...() e HistoryOrderGet...().

Para as funções OrderGetInteger() e HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

Identificador	Descrição	Tipo
ORDER_TICKET	Bilhete da ordem. Um número exclusivo atribuído a cada ordem	long
ORDER_TIME_SETUP	Hora de configuração de uma ordem	datetime
ORDER_TYPE	Tipo de ordem	ENUM_ORDER_TYPE
ORDER_STATE	Estado de uma ordem	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Hora de expiração de uma ordem	datetime
ORDER_TIME_DONE	Hora de execução ou cancelamento de uma ordem	datetime
ORDER_TIME_SETUP_MSC	O tempo para colocar uma ordem de execução em milissegundos desde 01.01.1970	long
ORDER_TIME_DONE_MSC	Tempo de execução e cancelamento de ordens em milissegundos desde 01.01.1970	long
ORDER_TYPE_FILLING	Type de preenchiment	ENUM_ORDER_TYPE_FILLING

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	o de uma ordem	
ORDER_TYPE_TIME	tempo de duração de uma ordem	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	ID de um Expert Advisor que colocou a ordem (projetado para garantir que cada Expert Advisor coloque seu próprio número único)	long
ORDER_REASON	Razão ou origem para a colocação da ordem	ENUM_ORDER_REASON
ORDER_POSITION_ID	Identificador de posição que é definido para uma ordem tão logo ela é executada. Cada ordem executada resulta em uma operação que abre ou modifica uma posição já existente. O identificador desta exata posição é atribuída à ordem executada neste momento.	long
ORDER_POSITION_BY_ID	Identificador da posição	long

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	oposta para as ordens do tipo	

Para as funções OrderGetDouble() e HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

Identificador	Descrição	Tipo
ORDER_VOLUME_INITIAL	Volume inicial de uma ordem	double
ORDER_VOLUME_CURRENT	Volume corrente de uma ordem	double
ORDER_PRICE_OPEN	Preço especificado na ordem	double
ORDER_SL	Valor de Stop Loss	double
ORDER_TP	Valor de Take Profit	double
ORDER_PRICE_CURRENT	O preço corrente do ativo de uma ordem	double
ORDER_PRICE_STOPLIMIT	O preço de ordem Limit para uma ordem StopLimit	double

Para as funções OrderGetString() e HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

Identificador	Descrição	Tipo
ORDER_SYMBOL	Ativo (symbol) de uma ordem	string
ORDER_COMMENT	Comentário sobre a ordem	string

Ao enviar uma solicitação de negociação usando a função OrderSend(), algumas operações requerem a indicação do tipo da ordem. O tipo de uma ordem é especificado no campo *type* da estrutura especial MqlTradeRequest, e pode aceitar valores da enumeração ENUM_ORDER_TYPE.

© 2000-2024, MetaQuotes Ltd.

ENUM_ORDER_TYPE

Identificador	Descrição
ORDER_TYPE_BUY	Ordem de Comprar a Mercado
ORDER_TYPE_SELL	Ordem de Vender a Mercado
ORDER_TYPE_BUY_LIMIT	Ordem pendente Buy Limit
ORDER_TYPE_SELL_LIMIT	Ordem pendente Sell Limit
ORDER_TYPE_BUY_STOP	Ordem pendente Buy Stop
ORDER_TYPE_SELL_STOP	Ordem pendente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Ao alcançar o preço da ordem, uma ordem pendente Buy Limit é colocada no preço StopLimit
ORDER_TYPE_SELL_STOP_LIMIT	Ao alcançar o preço da ordem, uma ordem pendente Sell Limit é colocada no preço StopLimit
ORDER_TYPE_CLOSE_BY	Ordem de fechamento da posição oposta

Cada ordem tem um status que descreve o seu estado. Para obter informações, use OrderGetInteger() ou HistoryOrderGetInteger() com o modificador ORDER_STATE. Valores permitidos são armazenados na enumeração ENUM_ORDEM_STATE.

ENUM_ORDER_STATE

Identificador	Descrição
ORDER_STATE_STARTED	Ordem verificada, mas ainda não aceita pela corretora (broker)
ORDER_STATE_PLACED	Ordem aceita
ORDER_STATE_CANCELED	Ordem cancelada pelo cliente
ORDER_STATE_PARTIAL	Ordem executada parcialmente
ORDER_STATE_FILLED	Ordem executada completamente
ORDER_STATE_REJECTED	Ordem rejeitada
ORDER_STATE_EXPIRED	Ordem expirada
ORDER_STATE_REQUEST_ADD	Ordem está sendo registrada (aplicação para o sistema de negociação)
ORDER_STATE_REQUEST_MODIFY	Ordem está sendo modificada (alterando seus parâmetros)
ORDER_STATE_REQUEST_CANCEL	Ordem está sendo excluída (excluindo a partir do sistema de negociação)

© 2000-2024, MetaQuotes Ltd.

Ao enviar uma ordem de negociação para ser executada **no momento atual** (time in force) é necessário especificar o preço e o volume de compra/venda. Ao fazer isso, vale a pena recordar que nos mercados financeiros não existe garantia de que num dado momento para dado ativo financeiro todo o volume solicitado estará disponível ao preço pretendido. É por esse motivo que a realização de operações de negociação em tempo real é regulada pelos modos de execução com base no preço e no volume. Os modos, ou seja, as políticas de execução, definem as regras para os casos em que o preço mudou ou o volume solicitado não pode ser totalmente cumprido **no momento**.

Modo de execução do preço - pode ser obtido na propriedade do símbolo SYMBOL_TRADE_EXEMODE que contém uma combinação de flags a partir da enumeração ENUM_SYMBOL_TRADE_EXECUTION.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
Modo de execução (Request Execution)	Execução da ordem de mercado ao preço previamente recebido da corretora. Antes de uma ordem de mercado ser enviada, são solicitados à corretora os preços pelos qual ela será executada. Após serem recebidos, a execuç	SYMBOL_TRADE_EXECUTION_REQUEST

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	ão da ordem a um determinado preço pode ser confirmada ou rejeitada.	
Execução imediata (Instant Execution)	Execução imediata de uma ordem de mercado ao preço especificado. Ao enviar uma ordem de negociação para ser executada, a plataforma substitui automaticamente os preços atuais na ordem.	SYMBOL_TRADE_EXECUTION_INSTANT

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	• Se a corretora aceitar o preço, a ordem será executada. • Se	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	a corretora não aceita a proposta, solicitado, ocorre a chamada	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	a da " recotação" (Requote) - a corretora retorna os preço	

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	sa os qu ais ao rd em po de se re xe cu ta da.	
Execução a mercado (Market Execution)	A decisão sobre o preço de execução é feita pela corretora sem acordo adicional com	SYMBOL_TRADE_EXECUTION_MARKET

© 2000-2024, MetaQuotes Ltd.

Modo de execução	Descrição	Valor em ENUM_SYMBOL_TRADE_EXECUTION
	o trader. O envio de uma ordem a mercado nesta modalidade implica que se aceita antecipadamente o preço a que será executada.	
Execução de bolsa (Exchange Execution)	As operações de negociação são realizadas aos preços das cotações atuais de mercado.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Política de preenchimento de volume– especificada na propriedade ORDER_TYPE_FILLING da ordem e só pode conter valores da enumeração ENUM_ORDER_TYPE_FILLING

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
Tudo/Nada (Fill or Kill)	A ordem	ORDER_FILLING_FOK

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	pode ser executada apenas com o volume especificado. Se não houver volume do ativo financeiro no mercado não for suficiente no momento, a ordem não será executada. O volume necessário pode ser composto por diversas ofertas atualmente disponíveis no mercado.	

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	A possibilidade de usar ordens FOK é determinada no servidor de negociação.	
Tudo/Parcial (Immediate or Cancel)	O trader aceita executar a transação com o volume máximo disponível no mercado, dentro dos limites especificados na ordem. Se a execução completa não for possível, a ordem será execut	ORDER_FILLING_IOC

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	ada com o volume disponível, e o volume não executado da ordem será cancelado. A possibilidade de usar ordens IOC é determinada no servidor de negociação.	
Passiva (Book or Cancel)	Uma ordem BoC implica que o pedido só pode ser colocado no livro de ofertas e não pode ser execut	ORDER_FILLING_BOC

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	ado imediatamente. Se a ordem pode ser executada imediatamente quando colocada, ela é retirada. Na verdade, a política do COB garante que o preço da ordem colocada será pior do que o mercado atual. As ordens BOC são utilizadas para negoci	

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	ação passiva, garantindo que a ordem não seja executada imediatamente após a colocação, e que não afete a liquidez atual. Ela é suportada somente para ordens limite e stop-limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER	

© 2000-2024, MetaQuotes Ltd.

Política de execução	Descrição	Valor em ENUM_ORDER_TYPE_FILLING
	_TYPE_SELL_STOP_LIMIT).	
Retornar (Return)	Em caso de execução parcial, a ordem com o volume residual continua em vigor, em vez de ser removida. Ordens Return não são permitidas no modo Market Execution (execução de ordens a mercado - SYMBOL_TRADE_EXECUTION_MARKET).	ORDER_FILLING_RETURN

© 2000-2024, MetaQuotes Ltd.

Ao enviar a ordem de negociação usando a função OrderSend(), a devida política de execução quanto ao volume pode ser definida no campo *type_filling* na estrutura especial MqlTradeRequest, com valores válidos da enumeração ENUM_ORDER_TYPE_FILLING. Para obter o valor desta propriedade em uma ordem específica ativa/concluída, use a função OrderGetInteger() ou HistoryOrderGetInteger() com o modificador ORDER_TYPE_FILLING.

Antes de enviar uma ordem para ser executada no momento atual, para a correta configuração do valor ORDER_TYPE_FILLING (tipo de execução por volume), para cada ativo financeiro por meio da função SymbolInfoInteger() você pode obter o valor da propriedade SYMBOL_FILLING_MODE que mostra os tipos de preenchimento por volume (permitidos para este símbolo) como uma combinação de flags. Deve-se observar que o tipo de preenchimento ORDER_FILLING_RETURN é sempre permitido, exceto para o modo "Execução da mercado" (SYMBOL_TRADE_EXECUTION_MARKET).

O uso de tipos de preenchimento dependendo do modo de execução pode ser representado na forma de uma tabela:

Modo de execução\Política de preenchimento	Tudo/Nada (FOK ORDER_FILLING_FOK)	Tudo/Parcial (IOC ORDER_FILLING_IOC)	Retornar (Return ORDER_FILLING_RETURN)
Execução imediata (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (independentemente da configuração do ativo)	+ (independentemente da configuração do ativo)	+ (sempre)
Execuções a pedido SYMBOL_TRADE_EXECUTION_REQUEST	+ (independentemente da configuração do ativo)	+ (independentemente da configuração do ativo)	+ (sempre)
Execução a mercado SYMBOL_TRADE_EXECUTION_MARKET	+ (definido nas configurações do ativo)	+ (definido nas configurações do ativo)	- (proibido independentemente da configuração do ativo)
Execução de bolsa SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (definido nas configurações do ativo)	+ (definido nas configurações do ativo)	+ (sempre)

Para ordens pendentes, independentemente do modo de execução (SYMBOL_TRADE_EXEMODE) é necessário utilizar o tipo de preenchimento ORDER_FILLING_RETURN, uma vez que essas ordens são executadas no momento do envio. Ao usar ordens pendentes, o trader aceita com antecedência que quando as condições para realização da transação nesta ordem forem atendidas, a corretora usará o tipo de preenchimento que é suportado por dada plataforma de negociação.

© 2000-2024, MetaQuotes Ltd.

O período de validade de uma ordem pode ser definida no campo *type_time* da estrutura especial MqlTradeRequest ao enviar uma solicitação de negociação usando a função OrderSend(). Valores da enumeração ENUM_ORDER_TYPE_TIME enumeration são permitidos. Para obter o valor desta propriedade use a função OrderGetInteger() ou HistoryOrderGetInteger() com o modificador ORDER_TYPE_TIME.

ENUM_ORDER_TYPE_TIME

Identificador	Descrição
ORDER_TIME_GTC	Ordem válida até cancelamento
ORDER_TIME_DAY	Ordem válida até o final do dia corrente de negociação
ORDER_TIME_SPECIFIED	Ordem válida até expiração
ORDER_TIME_SPECIFIED_DAY	A ordem permanecerá efetiva até 23:59:59 do dia especificado. Se esta hora está fora de uma sessão de negociação, a ordem expira na hora de negociação mais próxima.

A propriedade ORDER_REASON contém a razão para a colocação da ordem. A ordem pode ser colocada usando um programa MQL5, a partir de um aplicativo móvel ou como resultado de um evento StopOut, etc. Os valores possíveis ORDER_REASON são descritos na enumeração ENUM_ORDER_REASON.

ENUM_ORDER_REASON

Identificador	Descrição
ORDER_REASON_CLIENT	Ordem colocada a partir de um terminal desktop
ORDER_REASON_MOBILE	Ordem colocada a partir de um aplicativo móvel
ORDER_REASON_WEB	Ordem colocada a partir da plataforma web
ORDER_REASON_EXPERT	Ordem colocada a partir de um programa MQL5, Expert Advisor ou script
ORDER_REASON_SL	Ordem colocada como resultado da ativação do Stop Loss
ORDER_REASON_TP	Ordem colocada como resultado da ativação do Take Profit
ORDER_REASON_SO	Ordem colocada como resultado do evento Stop Out

© 2000-2024, MetaQuotes Ltd.

Propriedades de uma Posição

A execução de operações de negociação resulta na abertura de uma posição, alteração de seu volume, e/ou direção, ou no seu desaparecimento . Operações de negociação são conduzidos baseados em ordens, enviados pela função OrderSend() na forma de solicitações de negociação. Para cada ativo financeiro (symbol) apenas uma posição em aberto é possível. Uma posição tem um conjunto de propriedades disponíveis para leitura através de funções PositionGet...().

Para a função PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

Identificador	Descrição	Tipo
POSITION_TICKET	Bilhete da posição. Um número exclusivo atribuído a cada posição. Geralmente, ele corresponde ao bilhete da ordem, segundo o qual a posição foi aberta, exceto nos casos em que as operações de serviço no servidor tenham alterado o bilhete da ordem. Por exemplo, quando os swaps se acumulam com a reabertura de uma posição. Para localizar a ordem, segundo a qual foi aberta a posição, você deve utilizar a propriedade POSITION_IDENTIFIER. O valor da POSITION_TICKET corresponde a MqlTradeRequest::position.	long
POSITION_TIME	Hora de abertura de uma posição	datetime

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
POSITION_TIME_MSC	Posição de tempo de abertura em milissegundos desde 01.01.1970	long
POSITION_TIME_UPDATE	Posição de tempo de alteração	datetime
POSITION_TIME_UPDATE_MSC	Posição de tempo de alteração em milissegundo desde 01.01.1970	long
POSITION_TYPE	Tipo de posição	ENUM_POSITION_TYPE
POSITION_MAGIC	Número mágico de uma posição (veja ORDER_MAGIC)	long
POSITION_IDENTIFIER	Identificador de posição é um número único que é atribuído para toda nova posição aberta e não se altera durante todo o tempo de vida da posição. Movimentações de uma posição não alteram seu identificador. O identificador de posição é indicado em cada ordem (ORDER_POSITION_ID) e transação usada para abrir, alterar ou fechá-la. Utilize esta propriedade para procurar ordens e transações associadas com a posição. Durante a reversão de uma	long

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
	posição no modo de compensação (usando uma única transação de saída/entrada) o identificador de posição POSITION_IDENTIFIER não é alterado. No entanto, o POSITION_TICKET é substituído pelo bilhete de ordem, que levou à reversão. No modo de cobertura, não é fornecida a reversão de posição.	
POSITION_REASON	Razão para a abertura da posição	ENUM_POSITION_REASON

Para a função PositionGetDouble()

ENUM_POSITION_PROPERTY_DOUBLE

Identificador	Descrição	Tipo
POSITION_VOLUME	Volume de uma posição	double
POSITION_PRICE_OPEN	Preço de abertura de uma posição	double
POSITION_SL	Nível de Stop Loss de uma posição aberta	double
POSITION_TP	Nível de Take Profit de uma posição aberta	double
POSITION_PRICE_CURRENT	Preço corrente do ativo de uma posição	double
POSITION_SWAP	Swap acumulativo	double
POSITION_PROFIT	Lucro corrente	double

© 2000-2024, MetaQuotes Ltd.

Para a função PositionGetString()

ENUM_POSITION_PROPERTY_STRING

Identificador	Descrição	Tipo
POSITION_SYMBOL	Ativo (symbol) da posição	string
POSITION_COMMENT	Comentário de uma posição	string
POSITION_EXTERNAL_ID	ID de posição no sistema externo de negociação (na bolsa de valores)	string

A direção de uma posição em aberto (comprada ou vendida) é definida através do valor proveniente da enumeração ENUM_POSITION_TYPE. A fim de obter o tipo de uma posição em aberto use a função PositionGetInteger() com o modificador POSITION_TYPE.

ENUM_POSITION_TYPE

Identificador	Descrição
POSITION_TYPE_BUY	Compra
POSITION_TYPE_SELL	Venda

A propriedade POSITION_REASON contém a razão para a colocação da posição. A posição pode ser aberta como resultado da ativação de uma ordem colocada a partir de um terminal desktop, aplicativo móvel ou usando um Expert Advisor, etc. Os valores possíveis POSITION_REASON são descritos na enumeração ENUM_POSITION_REASON.

ENUM_POSITION_REASON

Identificador	Descrição
POSITION_REASON_CLIENT	Posição aberta como resultado da ativação de uma ordem colocada a partir de um terminal desktop
POSITION_REASON_MOBILE	Posição aberta como resultado da ativação de uma ordem colocada a partir de um aplicativo móvel
POSITION_REASON_WEB	Posição aberta como resultado da ativação de uma ordem colocada a partir da plataforma web
POSITION_REASON_EXPERT	Posição aberta como resultado da ativação de uma ordem colocada a partir de um programa MQL5, Expert Advisor ou script

© 2000-2024, MetaQuotes Ltd.

Propriedades da Transação

Uma transação é o reflexo de uma execução de uma negociação baseada em uma ordem que contém uma solicitação de negociação. Cada negociação é descrita através de propriedades que permitem obter informações sobre ela. A fim de ler valores destas propriedades, as funções do tipo HistoryDealGet...() são usadas, que retornam valores provenientes das enumerações correspondentes.

Para a função PlotIndexSetDouble()

ENUM_DEAL_PROPERTY_INTEGER

Identificador	Descrição	Tipo
DEAL_TICKET	Bilhete da transação. Um número exclusivo atribuído a cada transação.	long
DEAL_ORDER	Número da ordem da transação.	long
DEAL_TIME	Propriedades da Transação	datetime
DEAL_TIME_MSC	O tempo de execução de negociações em milissegundos desde 01.01.1970	long
DEAL_TYPE	Tipo de transação	ENUM_DEAL_TYPE
DEAL_ENTRY	Entrada de uma transação - entrada, saída, reversão	ENUM_DEAL_ENTRY
DEAL_MAGIC	Número mágico de uma transação (veja ORDER_MAGIC)	long
DEAL_REASON	Razão ou origem para realizar a transação	ENUM_DEAL_REASON
DEAL_POSITION_ID	Identificador de uma posição, na abertura, modificação ou fechamento da qual esta transação tomou parte. Cada posição tem um identificador único que é atribuído a todas as operações (deals) executadas para o ativo durante todo o tempo de vida da posição.	long

Para a função HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

Identificador	Descrição	Tipo
DEAL_VOLUME	Volume da transação	double
DEAL_PRICE	Preço da transação	double
DEAL_COMMISSION	Comissão da transação	double
DEAL_SWAP	Swap acumulativo no fechamento	double
DEAL_PROFIT	Lucro da transação	double

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
DEAL_FEE	O pagamento da transação é cobrado imediatamente após a ela ser concluída.	double
DEAL_SL	Nível de Stop Loss • Para uma operação de entrada ou reversão, o valor Stop Loss é tomado da ordem que abre ou reverte a posição • Para uma operação de saída, o valor do Stop Loss é tomado do fechamento da posição	double
DEAL_TP	Nível Take Profit • Para uma operação de entrada ou reversão, o valor Take Profit é tomado da ordem que abre ou reverte a posição • Para uma operação de saída, o valor Take Profit é tomado do fechamento da posição	double

Para a função HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

Identificador	Descrição	Tipo
DEAL_SYMBOL	Ativo (symbol) da transação	string
DEAL_COMMENT	Comentário da transação	string

Cada transação é caracterizada por um tipo, cujos valores permitidos são enumerados em ENUM_DEAL_TYPE. A fim de obter informações sobre o tipo de transação, use a função HistoryDealGetInteger() com o modificador DEAL_TYPE.

ENUM_DEAL_TYPE

Identificador	Descrição
DEAL_TYPE_BUY	Compra
DEAL_TYPE_SELL	Venda
DEAL_TYPE_BALANCE	Saldo
DEAL_TYPE_CREDIT	Crédito

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
DEAL_TYPE_CHARGE	Cobrança adicional
DEAL_TYPE_CORRECTION	Correção
DEAL_TYPE_BONUS	Bonus
DEAL_TYPE_COMMISSION	Comissão adicional
DEAL_TYPE_COMMISSION_DAILY	Comissão diária
DEAL_TYPE_COMMISSION_MONTHLY	Comissão mensal
DEAL_TYPE_COMMISSION_AGENT_DAILY	Comissão de agente diário
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Comissão de agente mensal
DEAL_TYPE_INTEREST	Taxa de juros
DEAL_TYPE_BUY_CANCELED	Operação de compra cancelada. Pode haver uma situação quando uma operação de compra executada anteriormente é cancelada. Neste caso, o tipo de transação executada anteriormente (DEAL_TYPE_BUY) é alterada para DEAL_TYPE_BUY_CANCELED, e seu lucro/prejuízo é zerado Lucro/prejuízo obtido anteriormente é cobrado/sacado usando uma operação de saldo separada
DEAL_TYPE_SELL_CANCELED	Operação de venda cancelada. Pode haver uma situação quando uma operação de venda executada anteriormente é cancelada. Neste caso, o tipo da operação executada anteriormente (DEAL_TYPE_SELL) é alterada para DEAL_TYPE_SELL_CANCELED, e seu lucro/prejuízo é zerado. Lucro/prejuízo obtido anteriormente é cobrado/sacado usando uma operação de saldo separada
DEAL_DIVIDEND	Operação de dividendos
DEAL_DIVIDEND_FRANKED	Operação de dividendos franqueados (não tributáveis)
DEAL_TAX	Cálculo do imposto

As operações (deal) diferem entre si não somente no seu conjunto de tipos em ENUM_DEAL_TYPE, mas também na forma como elas alteram posições. Isto pode ser uma simples abertura de posição, ou acumulação de uma posição aberta anteriormente (entrada de mercado), encerramento de posição através de uma operação oposta no volume correspondente (saída de mercado), ou reversão de posição, se a operação em direção oposta cobrir o volume da posição aberta anteriormente.

Todas estas situações são descritas pelos valores provenientes da enumeração ENUM_DEAL_ENTRY. A fim de receber esta informação sobre uma transação, use a função HistoryDealGetInteger() com o modificador DEAL_ENTRY.

ENUM_DEAL_ENTRY

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
DEAL_ENTRY_IN	Entrada
DEAL_ENTRY_OUT	Saída
DEAL_ENTRY_INOUT	Reversão
DEAL_ENTRY_OUT_BY	Fechamento pela posição oposta

A propriedade DEAL_REASON contém a razão para a realização da transação. A transação pode ser realizada como resultado da ativação da ordem colocada a partir do aplicativo móvel ou a partir de um programa MQL5; ou como resultado de um evento StopOut, ou debitada/creditada a margem de variação, etc. Os valores possíveis DEAL_REASON são descritos na enumeração ENUM_DEAL_REASON. Para as operações não-comerciais resultantes da alteração do balanço, crédito comissões e outras operações, indica-se como a razão DEAL_REASON_CLIENT.

ENUM_DEAL_REASON

Identificador	Descrição
DEAL_REASON_CLIENT	Transação realizada como resultado da ativação de uma ordem colocada a partir de um terminal desktop
DEAL_REASON_MOBILE	Transação realizada como resultado da ativação de uma ordem colocada a partir de um aplicativo móvel
DEAL_REASON_WEB	Transação realizada como resultado da ativação de uma ordem colocada a partir da plataforma web
DEAL_REASON_EXPERT	Transação realizada como resultado da ativação de uma ordem colocada a partir de um programa MQL5, Expert Advisor ou script
DEAL_REASON_SL	Transação realizada como resultado da ativação de uma ordem Stop Loss
DEAL_REASON_TP	Transação realizada como resultado da ativação de uma ordem Take Profit
DEAL_REASON_SO	Transação realizada como resultado do evento Stop Out
DEAL_REASON_ROLLOVER	Transação realizada devido à transferência da posição
DEAL_REASON_VMARGIN	Transação realizada após creditada/debitada a margem de variação
DEAL_REASON_SPLIT	Transação realizada após o fracionamento (redução do preço) do instrumento que tinha a posição aberta durante o fracionamento

© 2000-2024, MetaQuotes Ltd.

Tipos de Operações de Negociação (trade, transações, negociações)

A negociação realiza-se usando a função OrderSend()e enviando ordens não só para abrir posições, mas também para colocar, modificar e excluir ordens pendentes. Cada ordem de negociação contem uma indicação do tipo de operação solicitada. As operações de negociação são descritas na enumeração ENUM_TRADE_REQUEST_ACTIONS.

ENUM_TRADE_REQUEST_ACTIONS

Identificador	Descrição
TRADE_ACTION_DEAL	Coloca uma ordem de negociação para a transação ser executada imediatamente usando os parâmetros especificados (ordem de mercado)
TRADE_ACTION_PENDING	Coloca uma ordem de negociação para a transação ser executada sob certas condições (ordem pendente)
TRADE_ACTION_SLTP	Modifica valores de Stop-Loss e Take-Profit numa posição aberta
TRADE_ACTION_MODIFY	Modifca os parâmetros de uma ordem colocada anteriormente
TRADE_ACTION_REMOVE	Exclui uma ordem pendente colocada anteriormente
TRADE_ACTION_CLOSE_BY	Fechar a posição oposta

Exemplo de operação de negociação TRADE_ACTION_DEAL para abrir uma posição Buy:

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Abertura da posição Buy                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Atualização e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros do pedido
   request.action   =TRADE_ACTION_DEAL;                    // tipo de operação de neg
   request.symbol   =Symbol();                             // símbolo
   request.volume   =0.1;                                  // volume de 0.1 lotes
   request.type     =ORDER_TYPE_BUY;                       // tipo de ordem
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // preço para abertura
   request.deviation=5;                                    // desvio permitido do pre
   request.magic    =EXPERT_MAGIC;                         // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se não for possível env
//--- informação sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Exemplo de operação de negociaçãoTRADE_ACTION_DEAL para abertura da posição Sell:

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Abertura da posição Sell                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Declaração e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros do pedido
   request.action   =TRADE_ACTION_DEAL;                    // tipo de operação de neg
   request.symbol   =Symbol();                             // símbolo
   request.volume   =0.2;                                  // volume de 0.2 lotes
   request.type     =ORDER_TYPE_SELL;                      // tipo da ordem
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // preço para a abertura
   request.deviation=5;                                    // desvio permitido do pre
   request.magic    =EXPERT_MAGIC;                         // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se não for possível env
//--- informação sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Exemplo de ordem de negociação TRADE_ACTION_DEAL para fechadura de posições:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Fechadura de todas as posições                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaração do pedido e o seu resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posições abertas
//--- iterar todas as posições abertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parâmetros da ordem
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- saída de informação sobre a posição
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- se o MagicNumber coincidir
      if(magic==EXPERT_MAGIC)
        {
         //--- zerar os valores do pedido e os seus resultados
         ZeroMemory(request);
         ZeroMemory(result);
         //--- configuração dos parâmetros da ordem
         request.action   =TRADE_ACTION_DEAL;        // tipo de operação de negociação
         request.position =position_ticket;          // bilhete da posição
         request.symbol   =position_symbol;          // símbolo
         request.volume   =volume;                   // volume da posição
         request.deviation=5;                        // desvio permitido do preço
         request.magic    =EXPERT_MAGIC;             // MagicNumber da posição
         //--- colocação do preço e tipo de ordem dependendo do tipo de ordem
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- saída de informação sobre o fechamento
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- envio do pedido
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se não for possível
         //--- informação sobre a operação
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2024, MetaQuotes Ltd.

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

Exemplo de operação de negociaçãoTRADE_ACTION_PENDING para colocação de uma ordem pendente:

© 2000-2024, MetaQuotes Ltd.

```
#property description "Exemplo de colocação de ordens pendentes"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber do conselheiro
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // tipo da ordem
//+------------------------------------------------------------------+
//| Colocação de ordens pendentes                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- declaração e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros para colocação da ordem pendente
   request.action   =TRADE_ACTION_PENDING;                          // tipo de oper
   request.symbol   =Symbol();                                      // símbolo
   request.volume   =0.1;                                           // volume de 0
   request.deviation=2;                                             //desvio permit
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // deslocamento
   double price;                                                    // preço de exe
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // tamanho do p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // número de si
   //--- verificação do tipo de operação
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // tipo de orde
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // preço para a
      request.price    =NormalizeDouble(price,digits);             // нормализовав
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;    // цена для от
      request.price    =NormalizeDouble(price,digits);              // preço de ab
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // tipo de ord
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // preço de ab
      request.price=NormalizeDouble(price,digits);                 // preço de ab
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                      // tipo de ord
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // preço para
      request.price    =NormalizeDouble(price,digits);             // preço de ab
     }
   else Alert("Este exemplo é apenas para colocação de ordens pendentes");   // Se a o
//--- enviar a ordem
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());            // Se não for
//--- informações sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Exemplo da operação de negociação TRADE_ACTION_SLTP para alterar o valor Stop-Loss e Take-Profit na posição aberta:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Modificação do Stop-Loss e Take-Profit da posição                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaracão do pedido e o seu resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posições abertas
//--- iterar sobre todas as posições abertas
   for(int i=0; i<total; i++)
     {
      //--- parâmetros da ordem
      ulong  position_ticket=PositionGetTicket(i);// bilhete da posição
      string position_symbol=PositionGetString(POSITION_SYMBOL); // símbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // número d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber da posição
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume da posição
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss da posição
      double tp=PositionGetDouble(POSITION_TP);  // Take-Profit da posição
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- saída de informações sobre a posição
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se o MagicNumber coincidir, o Stop-Loss e o Take-Profit não estão definido
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2024, MetaQuotes Ltd.

```
         //--- cálculo dos níveis de preços atuais
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_I
         double price_level;
         //--- se o nível de deslocamento mínimo permitido em pontos a partir do preço
         if(stop_level<=0)
            stop_level=150; // definimos o deslocamento em 150 pontos a partir do prec
         else
            stop_level+=50; // definimos o nível de deslocamento (SYMBOL_TRADE_STOPS_I

         //--- cálculo e arredondamento dos valores Stop-Loss e Take-Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- zerar os valores de pedido e o seu resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- definição dos parâmetros de operação
         request.action  =TRADE_ACTION_SLTP; // tipo de operação de negociação
         request.position=position_ticket;   // bilhete da posição
         request.symbol=position_symbol;     // símbolo
         request.sl      =sl;                // Stop Loss da posição
         request.tp      =tp;                // Take Profit da posição
         request.magic=EXPERT_MAGIC;         // MagicNumber da posição
         //--- saída de informações sobre a modificação
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envio do pedido
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se não for possível
         //--- informações sobre a operação
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
      }
   }
//+------------------------------------------------------------------+
```

Exemplo de operação de negociação TRADE_ACTION_MODIFY para modificar os níveis de preços das ordens pendentes:

```
#define EXPERT_MAGIC 123456 // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Modificação de ordens pendentes                                  |
//+------------------------------------------------------------------+
void OnStart()
 {
//-- declaração e inicialização do pedido e o seu resultado
 MqlTradeRequest request={};
 MqlTradeResult result={};
 int total=OrdersTotal(); // número de ordens pendentes colocadas
//--- iterar todas as ordens pendentes colocadas
 for(int i=0; i<total; i++)
 {
 //--- parâmetros da ordem
 ulong order_ticket=OrderGetTicket(i); // bilhete da ordem
 string order_symbol=Symbol(); // símbolo
 int digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // número de signos de
 ulong magic=OrderGetInteger(ORDER_MAGIC); // MagicNumber da ordem
 double volume=OrderGetDouble(ORDER_VOLUME_CURRENT); // volume atual da ordem
 double sl=OrderGetDouble(ORDER_SL); // Stop Loss atual da ordem
 double tp=OrderGetDouble(ORDER_TP); // Take Profit atual da ordem
 ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo de ordem
 int offset = 50; // deslocação a partir do preço atual para colocação da ordem, em po
 double price; // preço de ativação da ordem
 double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT); // tamanho do ponto
 //--- saída de informações sobre a ordem
 PrintFormat("#%I64u %s %s %.2f %s sl: %s tp: %s [%I64d]",
 order_ticket,
 order_symbol,
 EnumToString(type),
 volume,
 DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
 DoubleToString(sl,digits),
 DoubleToString(tp,digits),
 magic);
 //--- se o MagicNumber coincidir, o Stop Loss e o Take Profit nao foram definidos
 if(magic==EXPERT_MAGIC && sl==0 && tp==0)
 {
 request.action=TRADE_ACTION_MODIFY; // tipo de operação de negociação
 request.order = OrderGetTicket(i); // bilhete da ordem
 request.symbol =Symbol(); // símbolo
 request.deviation=5; // desvio permitido a partir do preço
 //--- definição do nível de preços Take Profit e Stop Loss da ordem dependendo do seu
 if(type==ORDER_TYPE_BUY_LIMIT)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
 request.tp = NormalizeDouble(price+offset*point,digits);
 request.sl = NormalizeDouble(price-offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_SELL_LIMIT)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
 request.tp = NormalizeDouble(price-offset*point,digits);
 request.sl = NormalizeDouble(price+offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_BUY_STOP)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
 request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2024, MetaQuotes Ltd.

```
 request.sl = NormalizeDouble(price-offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_SELL_STOP)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
 request.tp = NormalizeDouble(price-offset*point,digits);
 request.sl = NormalizeDouble(price+offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 //--- envio do pedido
 if(!OrderSend(request,result))
 PrintFormat("OrderSend error %d",GetLastError()); // se não foi possível enviar o pe
 //--- informações sobre a operação
 PrintFormat("retcode=%u deal=%I64u order=%I64u",result.retcode,result.deal,result.or
 //--- zerado dos valores do pedido e o seu resultado
 ZeroMemory(request);
 ZeroMemory(result);
 }
 }
 }
//+------------------------------------------------------------------+
```

Exemplo de operação de negociação TRADE_ACTION_REMOVE para excluir ordens pendentes:

```
#define EXPERT_MAGIC 123456  // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Exclusão de ordens pendentes                                     |
//+------------------------------------------------------------------+
void OnStart()
 {
//-- Declaração e inicialização do pedido e resultado
 MqlTradeRequest request={};
 MqlTradeResult result={};
 int total=OrdersTotal(); // número de ordens pendentes colocadas
//--- iterar todas as ordens pendentes colocadas
 for(int i=total-1; i>=0; i--)
 {
 ulong order_ticket=OrderGetTicket(i); // bilhete da ordem
 ulong magic=OrderGetInteger(ORDER_MAGIC); // MagicNumber da ordem
 //--- se o MagicNumber coincidir
 if(magic==EXPERT_MAGIC)
 {
 //--- zerar os valores do pedido e o resultado
 ZeroMemory(request);
 ZeroMemory(result);
 //--- instalação de parâmetros de operação
 request.action=TRADE_ACTION_REMOVE; // tipo de operação de negociação
 request.order = order_ticket; // bilhete da ordem
 //--- envio do pedido
 if(!OrderSend(request,result))
 PrintFormat("OrderSend error %d",GetLastError()); // Se não for capaz de enviar o pe
 //--- informações sobre a operação
 PrintFormat("retcode=%u deal=%I64u order=%I64u",result.retcode,result.deal,result.or
 }
 }
 }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Exemplo de operação de negociação TRADE_ACTION_CLOSE_BY para fechamento da posição usando a oposta:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Fechamento de todas as posições opostas                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaração e inicialização do pedido e o seu resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // numero de posições abertas
//--- iterar sobre todas as posições abertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parâmetros da ordem
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      double sl=PositionGetDouble(POSITION_SL);
      double tp=PositionGetDouble(POSITION_TP);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- saída de informações sobre a posição
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se o MagicNumber coincidir
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // símbolo da posição oposta
            //--- se os símbolos das posições opostas e iniciais coincidirem
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- o tipo de posição oposta
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- saída, se os símbolos das posições opostas e iniciais coincidirem
               if(type==type_by)
                  continue;
               //--- zerar os valores do pedido e o seu resultado
               ZeroMemory(request);
               ZeroMemory(result);
               //--- definição os parâmetros de operação
               request.action=TRADE_ACTION_CLOSE_BY;                        // tipo d
               request.position=position_ticket;                            // bilhet
               request.position_by=PositionGetInteger(POSITION_TICKET);     // bilhet
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                                  // MagicN
               //--- saída de informações sobre o fechamento pela posição oposta
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- envio do pedido
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend error %d",GetLastError()); // se não for poss
```

```
                //--- informações sobre a operação
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Tipos de Transação de Negociação

Ao realizar algumas ações específicas em uma conta de negociação, seu estado muda. Tais ações includem:

- Envio de uma solicitação de negociação a partir de algum aplicativo MQL5 no terminal usando as funções OrderSend e OrderSendAsync e sua posterior execução.
- Envio de uma solicitação de negociação por meio da interface gráfica do terminal e sua posterior execução.
- Ativação de ordens pendentes e ordens de stop no servidor.
- Realização de operações no lado de um servidor de negociação.

As seguintes transações de negociação são realizadas como resultado destas ações:

- tratamento de uma solicitação de negociação
- mudança de ordens de abertura
- mudança de histórico de ordens
- mudança de histórico de operações (deals)
- mudança de posições

Por exemplo, ao enviar uma ordem de compra de mercado, ela é tratada, uma ordem de compra apropriada é criada para a conta, a ordem é então executada e removida da lista de ordens em aberto, e então ela é adicionada ao histórico de ordens, uma apropriada operação (deal) é adicionada ao histórico e uma nova posição é criada. Todas estas ações são transações de negociação.

Para deixar um programador rastrear as ações realizadas em relação a uma conta de negociação, a função OnTradeTransaction foi fornecida. Este handler permite obter transações de negociação aplicadas a uma conta em aplicações MQL5 A descrição de uma transação de negociação é submetida no primeiro parâmetro de OnTradeTransation usando a estrutura MqlTradeTransaction.

O tipo de transação de negociação é submetida no parâmetro de tipo de estrutura MqlTradeTransaction. Os possíveis tipos de transações de negociação são descritos através da seguinte enumeração:

ENUM_TRADE_TRANSACTION_TYPE

Identificador	Descrição
TRADE_TRANSACTION_ORDER_ADD	Adição de uma nova ordem de abertura.
TRADE_TRANSACTION_ORDER_UPDATE	Atualização de uma ordem de aberturar. As atualizações incluem não somente mudanças evidentes provenientes do terminal cliente ou do lado de um servidor de negociação, mas também alterações de estado de uma ordem ao configurá-lo (por exemplo, transição de ORDER_STATE_STARTED para ORDER_STATE_PLACED ou de ORDER_STATE_PLACED para ORDER_STATE_PARTIAL, etc.).
TRADE_TRANSACTION_ORDER_DELETE	Remoção de uma ordem da lista de ordens em aberto. Uma ordem pode ser excluída da lista de ordens em aberto como resultado da configuração de um

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	solicitação apropriada ou execução (preenchimento) ou movimentação para o histórico.
TRADE_TRANSACTION_DEAL_ADD	Adição de uma operação (deal) para o histórico. A ação é realizada como resultado de uma execução de uma ordem ou realização de operações com o saldo da conta.
TRADE_TRANSACTION_DEAL_UPDATE	Atualização de uma operação (deal) no histórico. Pode haver caso quando uma operação (deal) executada previamente é alterada em um servidor. Por exemplo, uma operação (deal) foi alterada em um sistema de negociação externo (exchange) onde ela foi transferida anteriormente por uma corretora (broker).
TRADE_TRANSACTION_DEAL_DELETE	Exclusão de uma operação (deal) do histórico. Pode haver casos quando uma operação (deal) executada anteriormente é excluída de um servidor. Por exemplo, uma operação (deal) foi excluída de um sistema de negociação externo (exchange) onde ela foi transferida anteriormente por uma corretora (broker).
TRADE_TRANSACTION_HISTORY_ADD	Adição de uma ordem no histórico como resultado de uma execução ou cancelamento.
TRADE_TRANSACTION_HISTORY_UPDATE	Alteração de uma ordem localizada no histórico de ordens. Este tipo é fornecido para aumentar a funcionalidade no lado de um servidor de negociação.
TRADE_TRANSACTION_HISTORY_DELETE	Exclusão de uma ordem do histórico de ordens. Este tipo é fornecido para aumentar a funcionalidade no lado de um servidor de negociação.
TRADE_TRANSACTION_POSITION	Alteração de uma posição não relacionada com a execução de uma operação (deal). Este tipo de transação mostra que uma posição foi alterada pelo lado de um servidor de negociação. O volume de uma posição, o preço de abertura, os níveis de Stop Loss e Take Profit podem ter sido alterados. Dados sobre alteração são submetidos na estrutura MqlTradeTransaction via o handler OnTradeTransaction. Alterações de posição (adição, alteração ou encerramento), como resultado de uma execução de operação (deal), não levam a ocorrência de uma transação TRADE_TRANSACTION_POSITION.
TRADE_TRANSACTION_REQUEST	Notificação do fato de que uma solicitação de negociação foi processada por um servidor e o resultado processado foi recebido. Apenas o campo type (tipo de transação de negociação) deve ser analisado em tais transações na estrutura

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	MqlTradeTransaction. O segundo e terceiro parâmetros de OnTradeTransaction (request e result) devem ser analisados para dados adicionais.

Dependendo do tipo da transação de negociação, vários parâmetros são preenchidos na estrutura MqlTradeTransaction, descrevendo-a. Uma descrição detalhada dos dados submetidos é mostrada em Estrutura de uma Transação de Negociação.

Também Veja

Estrutura de uma Transação de Negociação, OnTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Ordens de Negociação na Profundidade de Mercado

Para ações, a janela Profundidade de Mercado é disponível, onde você pode ver as ordens de Compra e Venda correntes. A direção desejada de uma operação de negociação, a quantidade requerida e o preço solicitado são especificados em cada ordem.

Para obter informações sobre o estado corrente do DOM por meio de MQL5, a função MarketBookGet() é usada, que coloca uma "fotografia instantânea" do DOM no array de estruturas MqlBookInfo. Cada elemento do array no campo *type* contém informações sobre a direção da ordem - o valor da enumeração ENUM_BOOK_TYPE.

ENUM_BOOK_TYPE

Identificador	Descrição
BOOK_TYPE_SELL	Ordem de venda (Offer)
BOOK_TYPE_BUY	Ordem de compra (Bid)
BOOK_TYPE_SELL_MARKET	Ordem de venda (Offer)
BOOK_TYPE_BUY_MARKET	Ordem de venda (Offer)

Também Veja

Estruturas e classes, Estruturas do DOM, Tipos de operações de negociação, Informações de Mercado

© 2000-2024, MetaQuotes Ltd.

Propriedades de Sinal

As seguintes enumerações são usadas ao trabalhar com sinais de negociação e configurações da cópia de sinal.

Enumeração de propriedades do tipo double do sinal de negociação:

ENUM_SIGNAL_BASE_DOUBLE

ID	Descrição
SIGNAL_BASE_BALANCE	Saldo da conta
SIGNAL_BASE_EQUITY	Saldo da conta
SIGNAL_BASE_GAIN	Ganho da conta
SIGNAL_BASE_MAX_DRAWDOWN	Retirada máxima da conta
SIGNAL_BASE_PRICE	Preço de assinatura do sinal
SIGNAL_BASE_ROI	Retorno do Investimento (%)

Enumeração de propriedades do tipo integer do sinal de negociação

ENUM_SIGNAL_BASE_INTEGER

ID	Descrição
SIGNAL_BASE_DATE_PUBLISHED	Data de publicação (data em que estará disponível para assinar)
SIGNAL_BASE_DATE_STARTED	Data de início da monitorização
SIGNAL_BASE_DATE_UPDATED	Data da última atualização da estatística de negociação do sinal
SIGNAL_BASE_ID	ID do sinal
SIGNAL_BASE_LEVERAGE	Alavancagem da conta
SIGNAL_BASE_PIPS	Lucro em pips
SIGNAL_BASE_RATING	Posição na classificação
SIGNAL_BASE_SUBSCRIBERS	Número de assinantess
SIGNAL_BASE_TRADES	Número de negociações
SIGNAL_BASE_TRADE_MODE	Tipo de conta (0-real, 1-demo, 2-concurso)

Enumeração de propriedades do tipo string do sinal de negociação:

ENUM_SIGNAL_BASE_STRING

ID	Descrição
SIGNAL_BASE_AUTHOR_LOGIN	Login do autor

© 2000-2024, MetaQuotes Ltd.

ID	Descrição
SIGNAL_BASE_BROKER	Nome da corretora (empresa)
SIGNAL_BASE_BROKER_SERVER	Server da corretora
SIGNAL_BASE_NAME	Nome do sinal
SIGNAL_BASE_CURRENCY	Moeda base do sinal

Enumeração de propriedades do tipo double das configurações de cópia do sinal:

ENUM_SIGNAL_INFO_DOUBLE

ID	Descrição
SIGNAL_INFO_EQUITY_LIMIT	Limite patrimonial
SIGNAL_INFO_SLIPPAGE	Deslizamento (usado ao colocar ordens de mercado em sincronização de posições e cópia de negociações)
SIGNAL_INFO_VOLUME_PERCENT	Porcentagem máxima do depósito utilizado (%), r/o

Enumeração de propriedades do tipo integer das configurações de cópia do sinal:

ENUM_SIGNAL_INFO_INTEGER

ID	Descrição
SIGNAL_INFO_CONFIRMATIONS_DISABLED	O sinalizador permite a sincronização sem caixa de diálogo de confirmação
SIGNAL_INFO_COPY_SLTP	Copia as flags Stop Loss e Take Profit
SIGNAL_INFO_DEPOSIT_PERCENT	Porcentagem de depósito (%)
SIGNAL_INFO_ID	ID do sinal, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	Permissão da flag "Copiar negociações por assinaturas"
SIGNAL_INFO_TERMS_AGREE	Flag "Concordar com os termos de uso do serviço de Sinais", r/o

Enumeração de propriedades do tipo string das configurações de cópia do sinal:

ENUM_SIGNAL_INFO_STRING

ID	Descrição
SIGNAL_INFO_NAME	Nome do sinal, r/o

Veja também

Sinais de Negociação

© 2000-2024, MetaQuotes Ltd.

Constantes Nomeados

Todas as constantes usadas em MQL5 podem ser divididas nos seguintes grupos:

- Substituições macro predefinidas – valores são substituídos durante compilação;
- Constantes matemáticas – valores de algumas expressões matemáticas;
- Constantes de tipo numérico – algumas restrições de tipo simples;
- Códigos de motivos de Desinicialização – descrição de motivos de não inicialização;
- Verificação de Ponteiro de Objeto – enumeração dos tipos de ponteiros retornados pela função CheckPointer();
- Outras constantes – todas as outras constantes.

© 2000-2024, MetaQuotes Ltd.

Substituições de Macro Predefinidas

Para simplificar o processo de depuração e obter informações sobre a operação de um programa-mql5, existem constantes de macro especiais, valores que são definidos no momento da compilação. A forma mais fácil de usar estas constantes é imprimir estes valores através da função Print(), como é mostrado no exemplo.

Constante	Descrição
__CPU_ARCHITECTURE__	Nome da arquitetura (conjunto de comandos) para a qual o arquivo EX5 é criado no momento da compilação
__DATE__	Data de compilação do arquivo sem a hora (horas, minutos e segundos são iguais a 0)
__DATETIME__	Data e hora de compilação do arquivo
__LINE__	Assinatura da função em cujo corpo a macro está localizada.
__FILE__	Nome do arquivo compilado corrente
__PATH__	O caminho absoluto para o arquivo que está sendo compilado correntemente
__FUNCTION__	Nome da função, em cujo corpo a macro está localizada
__FUNCSIG__	Assinatura da função em cujo corpo a macro está localizada. O registro da descrição completa de funções pode ser útil na identificação de funções sobrecarregadas
__MQLBUILD__, __MQL5BUILD__	Número do compilação
__COUNTER__	Para cada declaração de __COUNTER__ que encontra, o compilador substitui no local o valor do contador de 0 a N-1, onde N é o número de usos no código. Ao recompilar o código-fonte sem alterações, a ordem de __COUNTER__ é garantida. O valor do contador __COUNTER__ é calculado da seguinte forma: • o valor inicial do contador é 0, • após cada uso do contador, seu valor é aumentado em 1, • primeiro, o compilador expande todas as macros e modelos em código-fonte no local, • código separado é gerado para cada especialização da função baseada em modelo, • código separado é gerado para cada especialização da classe/estrutura baseada em modelo, • o compilador então percorre o código-fonte resultante usando a ordem definida e substitui cada uso de __COUNTER__ que encontra pelo valor do contador atual.

© 2000-2024, MetaQuotes Ltd.

Constante	Descrição
	No exemplo abaixo, você pode ver claramente como o compilador processa o código-fonte e substitui o __COUNTER__ encontrado com valores crescentes sequencialmente.
__RANDOM__	Para cada declaração de __RANDOM__, o compilador substitui um número aleatório ulong no código.

Exemplo:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- um exemplo de impressão de informação na inicialização de um Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- define o intervalo entre os eventos timer
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- um exemplo de impressão de informação na desinicialização de um Expert Advisor
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- impressão de informação no recebimento de um tick
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- impressão de informação dentro da função
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
//--- impressão de informação dentro da função
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
//| OnTimer event handler                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

Exemplo para aprender a trabalhar com uma macro __COUNTER__

```
//--- criamos uma macro para impressão rápida da expressão e seu valor no log
#define print(expr) Print(#expr,"=",expr)

//--- definimos uma macro customizada MACRO_COUNTER por meio da macro predefinida __C
#define MACRO_COUNTER __COUNTER__

//--- definimos o valor da variável input com a macro __COUNTER__
input int InpVariable = __COUNTER__;

//--- definimos o valor da variável global com a macro __COUNTER__ antes de definir as
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| a função retorna o valor __COUNTER__                             |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| função de modelo retorna valor __COUNTER__                       |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| estrutura com um método que retorna __COUNTER__                  |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // não é usado

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| estrutura baseada em modelo com um método que retorna __COUNTER__|
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // não é usado

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| estrutura com um método baseado em modelo que retorna __COUNTER__|
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // não é usado

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| função #2, que retorna o valor __COUNTER__                       |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart(void)
  {
// __COUNTER__ em macros e variáveis
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

/--- __COUNTER__ em funções
   print(GlobalFunc());
   print(GlobalFunc());                // o valor não muda
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());   // o valor não muda
   print(GlobalTemplateFunc<double>());// o valor mudou
   print(GlobalFunc2());
   print(GlobalFunc2());               // o valor não muda

// __COUNTER__ na estrutura
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                 // o valor não muda

// __COUNTER__ na estrutura baseada em modelo
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                 // o valor não muda
   print(b3.Method());                 // o valor mudou

// __COUNTER__ em uma estrutura com uma função baseada em modelo
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());         // o valor mudou
   print(c2.Method<int>());            // valor como ao chamar c1.Method<int>() pela p

//--- mais uma vez, olhamos para __COUNTER__ na macro e na variável global
   print(MACRO_COUNTER);  // o valor mudou
   print(ExtGlobal2);
  }
//--- definimos o valor da variável global com a macro __COUNTER__ após as definições
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Resultado
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
```

© 2000-2024, MetaQuotes Ltd.

```
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
   b3.Method()=11
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

© 2000-2024, MetaQuotes Ltd.

Constantes Matemáticas

Constantes especiais contendo valores que são reservados para algumas expressões matemáticas. Estas constantes podem ser usadas em qualquer lugar no programa ao invés de calcular seus valores usando funções matemáticas.

Constante	Descrição	Valor
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.434294481903251827651
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.785398163397448309616
M_1_PI	1/pi	0.318309886183790671538
M_2_PI	2/pi	0.636619772367581343076
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.707106781186547524401

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprime os valores das constantes
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

Constantes de Tipo Numérico

Cada tipo numérico simples é destinado para um certo tipo de tarefa e permite otimizar a operação de um programa-mql5 quando usado corretamente. Para uma melhor legibilidade de código e correto manejo dos resultados de cálculos, existem constantes que permitem receber informações sobre um conjunto de restrições definido a um certo tipo de dados simples.

Constante	Descrição	Valor
CHAR_MIN	Valor mínimo, que pode ser representado pelo tipo char	-128
CHAR_MAX	Valor máximo, que pode ser representado pelo tipo char	127
UCHAR_MAX	Valor máximo, que pode ser representado pelo tipo uchar	255
SHORT_MIN	Valor mínimo, que pode ser representado pelo tipo short	-32768
SHORT_MAX	Valor máximo, que pode ser representado pelo tipo short	32767
USHORT_MAX	Valor máximo, que pode ser representado pelo tipo ushort	65535
INT_MIN	Valor mínimo, que pode ser representado pelo tipo int	-2147483648
INT_MAX	Valor máximo, que pode ser representado pelo tipo int	2147483647
UINT_MAX	Valor máximo, que pode ser representado pelo tipo uint	4294967295
LONG_MIN	Valor mínimo, que pode ser representado pelo tipo long	-9223372036854775808
LONG_MAX	Valor máximo, que pode ser representado pelo tipo long	9223372036854775807
ULONG_MAX	Valor máximo, que pode ser representado pelo tipo ulong	18446744073709551615
DBL_MIN	Valor positivo mínimo, que pode ser representado pelo tipo double	2.2250738585072014e-308
DBL_MAX	Valor máximo, que pode ser representado pelo tipo double	1.7976931348623158e+308
DBL_EPSILON	Valor mínimo, que satisfaz a condição: 1.0+DBL_EPSILON != 1.0 (para o tipo double)	2.2204460492503131e-016
DBL_DIG	Número de algarismos decimais significativos para o tipo double	15

© 2000-2024, MetaQuotes Ltd.

Constante	Descrição	Valor
DBL_MANT_DIG	Quantidade de bits em uma mantissa para o tipo double	53
DBL_MAX_10_EXP	Valor decimal máximo de grau de expoente para o tipo double	308
DBL_MAX_EXP	Valor binário máximo de grau de expoente para o tipo double	1024
DBL_MIN_10_EXP	Valor decimal mínimo de grau de expoente para o tipo double	(-307)
DBL_MIN_EXP	Valor binário mínimo de grau de expoente para o tipo double	(-1021)
FLT_MIN	Valor positivo mínimo, que pode ser representado pelo tipo float	1.175494351e-38
FLT_MAX	Valor máximo, que pode ser representado pelo tipo float	3.402823466e+38
FLT_EPSILON	Valor mínimo, que satisfaz a condição: 1.0+DBL_EPSILON != 1.0 (para o tipo float)	1.192092896e-07
FLT_DIG	Número de algarismos decimais significativos para o tipo float	6
FLT_MANT_DIG	Quantidade de bits em uma mantissa para o tipo float	24
FLT_MAX_10_EXP	Valor decimal máximo de grau de expoente para o tipo float	38
FLT_MAX_EXP	Valor binário máximo de grau de expoente para o tipo float	128
FLT_MIN_10_EXP	Valor decimal mínimo de grau de expoente para o tipo float	-37
FLT_MIN_EXP	Valor binário mínimo de grau de expoente para o tipo float	(-125)

Exemplo:

```
void OnStart()
  {
//--- imprime os valores de constantes
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
```

© 2000-2024, MetaQuotes Ltd.

```
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

© 2000-2024, MetaQuotes Ltd.

Códigos de Motivos de Desinicialização

Os códigos de razão de não inicialização são devolvidos pela função UninitializeReason(). Os valores possíveis são os seguintes:

Constante	Valor	Descrição
REASON_PROGRAM	0	O Expert Advisor terminou sua operação chamando a função ExpertRemove()
REASON_REMOVE	1	O programa foi excluído do gráfico
REASON_RECOMPILE	2	O programa foi recompilado
REASON_CHARTCHANGE	3	O período do símbolo ou gráfico foi alterado
REASON_CHARTCLOSE	4	O gráfico foi encerrado
REASON_PARAMETERS	5	Os parâmetros de entrada foram alterados por um usuário
REASON_ACCOUNT	6	Outra conta foi ativada ou o servidor de negociação foi reconectado devido a alterações nas configurações de conta
REASON_TEMPLATE	7	Um novo modelo foi aplicado
REASON_INITFAILED	8	Esse valor significa que o manipulador OnInit() retornou um valor diferente de zero
REASON_CLOSE	9	Terminal foi fechado

O código de razão de desinicialização também é passado como um parâmetro da função predeterminada OnDeinit(const int reason).

Exemplo:

```
//+------------------------------------------------------------------+
//| obter descrição do texto                                         |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
```

© 2000-2024, MetaQuotes Ltd.

```
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- A primeira maneira de obter o código de razão de desinicialização
   Print(__FUNCTION__,"_Uninitalization reason code = ",reason);
    //--- A segunda maneira de obter o código de razão de desinicialização
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2024, MetaQuotes Ltd.

Verificação de Ponteiro de Objeto

A função CheckPointer() é usado para verificar o tipo do ponteiro de objeto. A função retorna um valor da enumeração ENUM_POINTER_TYPE. Caso um ponteiro incorreto seja usado, a execução do programa será imediatamente finalizada.

Objetos criados por meio do operador new() são do tipo POINTER_DYNAMIC. O operador delete() pode e deve ser usado somente para tais ponteiros.

Todos os outros ponteiros são do tipo POINTER_AUTOMATIC, que significa que este objeto foi criado automaticamente pelo ambiente do programa mql5. Tais objetos são excluídos automaticamente após serem usados.

ENUM_POINTER_TYPE

Constante	Descrição
POINTER_INVALID	Ponteiro incorreto
POINTER_DYNAMIC	Ponteiro do objeto criado por meio do operador new()
POINTER_AUTOMATIC	Ponteiro de qualquer objeto criado automaticamente (sem usar new())

Também Veja

Erros em Tempo de Execução, Exclusão de Objeto Operador delete, CheckPointer

© 2000-2024, MetaQuotes Ltd.

Outras Constantes

A constante CLR_NONE é usada para indicar a ausência de cor, ela significa que o objeto gráfico ou a série gráfica de um indicador não serão plotados. Esta constante não foi incluída na lista de constantes Web-color, mas ela pode ser aplicada em todo lugar onde os argumentos de cor forem requeridos.

A constante INVALID_HANDLE pode ser usada para verificar handles de arquivo (veja FileOpen() e FileFindFirst()).

Constante	Descrição	Valor
CHARTS_MAX	O número possível máximo de gráficos abertos simultaneamente em um terminal	100
clrNONE	Ausência de cor	-1
EMPTY_VALUE	Valor vazio em um buffer de indicador	DBL_MAX
INVALID_HANDLE	Manipulador incorreto	-1
IS_DEBUG_MODE	Flag que indica que um programa MQL5 opera em modo de depuração	não-zero para modo de depuração, zero caso contrário
IS_PROFILE_MODE	Flag que indica que um programa MQL5 opera em modo de profiling	não-zero para modo de profiling, zero caso contrário
NULL	Zero para quaisquer tipos	0
WHOLE_ARRAY	Significa o número de itens restantes até o final do array, isto é, o array inteiro será processado	-1
WRONG_VALUE	A constante pode ser implicitamente convertido para qualquer tipo de enumeração	-1

A constante EMPTY_VALUE geralmente corresponde aos valores de indicadores que não são exibidos no gráfico. Por exemplo, para o indicador incorporado Desvio Padrão (Standard Deviation) com um período de 20, a linha para as primeiras 19 barras no histórico não é exibida no gráfico. Se você criar um handle deste indicador com a função iStdDev() e copiar os valores do indicador destas barras para um array através de CopyBuffer(), então estes valores serão iguais a EMPTY_VALUE.

Você pode escolher especificar em um indicador customizado seus próprios valores vazios, indicando quando o indicador não deve ser desenhado no gráfico. Use a função PlotIndexSetDouble() com a modificador PLOT_EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

A constante NULL pode ser atribuído a uma variável de qualquer tipo simples ou a um objeto, estrutura ou ponteiro de classe. A atribuição NULL a uma variável string significa a completa desinicialização desta variável.

A constante WRONG_VALUE é destinada para casos em que é necessário retornar valor de uma enumeração, e ela deve ser um valor errado. Por exemplo, quando nós precisamos informar que um valor de retorno é um valor proveniente desta enumeração. Vamos considerar como um exemplo a função CheckLineStyle(), que retorna o estilo de linha para um objeto, especificado seu nome. Se na verificação de estilo por ObjectGetInteger() o resultado for true, um valor proveniente de ENUM_LINE_STYLE é retornado; caso contrário WRONG_VALUE é retornado.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Erro ao obter estilo de linha.");
  }
//+------------------------------------------------------------------+
//| retorna o estilo de linha para um obj. espec. atráves de seu nome|
//+------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

A constante WHOLE_ARRAY é destinada para funções que requerem especificar o número de elementos nos arrays processados:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Se você quiser especificar que todos os valores do array a partir de uma posição especificada até o final devem ser processados, você deve especificar apenas o valor WHOLE_ARRAY.

A constante IS_PROFILE_MODE permite alterar uma operação de programa para correta coleta de dados no modo de profiling. Profiling permite medir o tempo de execução de fragmentos individuais do programa (geralmente abrangendo funções), bem como calcular o número de tais chamadas. Chamadas a função Sleep() podem ser desabilitadas para determinar o tempo de execução no modo de profiling, como neste exemplo:

```
//--- Sleep pode afetar (alterar) enormemente o resultado do profiling
```

© 2000-2024, MetaQuotes Ltd.

```
if(!IS_PROFILE_MODE) Sleep(100); // desabilita a chamada Sleep() no modo de profiling
```

O valor da constante IS_PROFILE_MODE é definida pelo compilador durante a compilação, enquanto que ela é definida para zero no modo convencional. Ao iniciar um programa no modo de profiling, uma compilação especial é realizada e IS_PROFILE_MODE é substituída com um valor não-zero.

A constante IS_DEBUG_MODE pode ser útil quando se precisar alterar levemente a operação de um programa MQL5 no modo de depuração. Por exemplo, no modo de depuração você pode precisar exibir informações de depuração adicionais no log do terminal ou criar objetos gráficos adicionais em um gráfico.

O seguinte exemplo cria um objeto Label e define sua descrição e cor dependendo do modo de execução do script. A fim de executar um script em modo de depuração a partir do MetaEditor, pressione F5. Se você executar um script a partir da janela de navegação do terminal, então a cor e o texto do objeto Label serão diferentes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              Check_DEBUG_MODE.mq5 |
//|                      Copyright © 2009, MetaQuotes Software Corp. |
//|                                         https://www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Objeto",label_name,"não encontrado. Código de erro = ",GetLastError());
      //--- cria um Rótulo (Label)
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- define coordenada X
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- define coordenada Y
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // modo de depuração
        {
         //--- exibe mensagem sobre o modo de execução do script
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- defini a cor de texto para vermelho
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Não é possível definir a cor. Error",GetLastError());
        }
```

```
      else              // modo de operação
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- defini a cor de texto para vermelho
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Não é possível definir a cor. Error ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();    // aqui finalização ocorrerá, se nós estivermos em modo de depu
     }
  }
```

Crypt Methods

A enumeração ENUM_CRYPT_METHOD é usada para especificar o método de transformação de dados, usado nas funções CryptEncode() e CryptDecode().

ENUM_CRYPT_METHOD

Constant	Descrição
CRYPT_BASE64	BASE64
CRYPT_AES128	Criptografia AES com chave de 128 bits (16 bytes)
CRYPT_AES256	Criptografia AES com chave de 256 bits (32 bytes)
CRYPT_DES	Criptografia DES com chave de 56 bits (7 bytes)
CRYPT_HASH_SHA1	SHA1 HASH caculation
CRYPT_HASH_SHA256	SHA256 HASH caculation
CRYPT_HASH_MD5	MD5 HASH caculation
CRYPT_ARCH_ZIP	ZIP archives

Também Veja

DebugBreak, Propriedades de programas MQL5 executados, CryptEncode(), CryptDecode()

© 2000-2024, MetaQuotes Ltd.

Estruturas de Dados

A linguagem MQL5 oferece 12 estruturas predefinidas:

- MqlDateTime é destinado para trabalhar com data e hora;
- MqlParam pode enviar parâmetros de entrada quando criar um indicador usando a função IndicatorCreate();
- MqlRates é destinado para manipular dados históricos, ela contém informações sobre o preço, o volume e o spread;
- MqlBookInfo é destinado para obter informações sobre a O livro de ofertas;
- MqlTradeRequest é usado para criar uma solicitação para operações de negociação;
- MqlTradeCheckResult permite verificar se a solicitação de negociação está prestes antes de ser enviada;
- MqlTradeResult contém a resposta do servidor de negociação a uma solicitação de negociação, enviada via função OrderSend();
- MqlTradeTransaction contém a descrição de uma operação de negociação;
- MqlTick é projetado para recuperar rapidamente a maioria das informações requeridas sobre os preços atuais.
- Estruturas do Calendário Econômico são destinadas para obter informações sobre os eventos do Calendário econômico que entram na plataforma MetaTrader 5 em tempo real. Funções do Calendário econômico permitem analisar indicadores macroeconômicos imediatamente após a publicação de novos relatórios, uma vez que os valores reais são transmitidos diretamente da fonte sem atraso.

© 2000-2024, MetaQuotes Ltd.

MqlDateTime

A estrutura de tipo data contém oito campos do tipo int:

```
struct MqlDateTime
  {
   int year;            // Ano
   int mon;             // Mês
   int day;             // Dia
   int hour;            // Hora
   int min;             // Minutos
   int sec;             // Segundos
   int day_of_week;     // Dia da semana (0-domingo, 1-segunda, ... ,6-sábado)
   int day_of_year;     // Número do dia do ano (1 de Janeiro é atribuído o valor 0)
  };
```

Observação

O número do dia do ano, day_of_year, para ano bissexto, a partir de março, diferirá do número do dia correspondente a um ano não bissexto.

Exemplo:

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Resultado:
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

Também Veja

TimeToStruct, Estruturas e Classes

© 2000-2024, MetaQuotes Ltd.

A Estrutura de Entrada de Parâmetros de Indicadores (MqlParam)

A estrutura MqlParam foi projetada especialmente para fornecer parâmetros de entrada quando criar um um indicador técnico usando a função IndicatorCreate().

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                    // tipo de parâmetro de entrada, valor d
   long              integer_value;           // campo para armazenar um tipo inteiro
   double            double_value;            // campo para armazenar um tipo double
   string            string_value;            // campo para armazenar um tipo string
  };
```

Todos os parâmetros de entrada de um indicador são transmitidos na forma de um array do tipo MqlParam, o campo *type* de cada elemento deste array especifica o tipo de dado transmitido pelo elemento. Os valores do indicador devem ser primeiro colocados nos campos apropriados para cada elemento (in *integer_value*, in *double_value* or *string_value*) dependendo de qual valor da enumeração ENUM_DATATYPE é especificado no campo *type*.

Se o valor IND_CUSTOM for passado como tipo do indicador para função IndicatorCreate(), o primeiro elemento do array de parâmetros de entrada deve ter o campo *type* com o valor de TYPE_STRING proveniente da enumeração ENUM_DATATYPE, e o campo *string_value* deve conter o nome do indicador customizado.

© 2000-2024, MetaQuotes Ltd.

MqlRates

Esta estrutura armazena informações sobre os preços, volumes e spread.

```
struct MqlRates
  {
   datetime time;         // Hora inicial do período
   double   open;         // Preço de abertura
   double   high;         // O preço mais alto do período
   double   low;          // O preço mais baixo do período
   double   close;        // Preço de fechamento
   long     tick_volume;  // Volume de Tick
   int      spread;       // Spread
   long     real_volume;  // Volume de negociação
  };
```

Exemplo:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Erro ao copiar dados de preços ",GetLastError());
   else Print("Copied ",ArraySize(rates)," bars");
  }
```

Também Veja

CopyRates, Acesso a séries de tempo

© 2000-2024, MetaQuotes Ltd.

MqlBookInfo

Fornece informações sobre os dados de profundidade de mercado.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE   type;          // Tipo de ordem proveniente da enumeração ENUM_B
   double           price;         // Preço
   long             volume;        // Volume
   double           volume_real;   // Volume com maior precisão
  };
```

Observação

A estrutura MqlBookInfo é predefinida, portanto ela não requer nenhuma declaração ou descrição. Para usar a estrutura, apenas declare a variável deste tipo.

O DOM (Profundidade de Mercado) é disponível somente para alguns ativos (symbols).

Exemplo:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo sobre ",Symbol());
     }
   else
     {
      Print("Falha ao receber DOM para o símbolo ",Symbol());
     }
```

Também Veja

MarketBookAdd, MarketBookRelease, MarketBookGet, Ordens de Negociação em DOM, Tipos de Dados

© 2000-2024, MetaQuotes Ltd.

A Estrutura de Solicitação de Negociação (MqlTradeRequest)

A interação entre o terminal do cliente e o servidor de negociação para colocar ordens realiza-se usando solicitações de negociação (pedidos de negociação). A solicitação de negociação é apresentada por uma estrutura predefinida especial chamada de MqlTradeRequest, ela contém todos os campos necessários para realizar operações (deals) de negociação. O resultado do processamento da solicitação é apresentado pela mesma estrutura MqlTradeResult.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;          // Tipo de operação de negociação
   ulong                         magic;           // Expert Advisor -conselheiro- ID
   ulong                         order;           // Bilhetagem da ordem
   string                        symbol;          // Símbolo de negociação
   double                        volume;          // Volume solicitado para uma encor
   double                        price;           // Preço
   double                        stoplimit;       // Nível StopLimit da ordem
   double                        sl;              // Nível Stop Loss da ordem
   double                        tp;              // Nível Take Profit da ordem
   ulong                         deviation;       // Máximo desvio possível a partir
   ENUM_ORDER_TYPE               type;            // Tipo de ordem
   ENUM_ORDER_TYPE_FILLING       type_filling;    // Tipo de execução da ordem
   ENUM_ORDER_TYPE_TIME          type_time;       // Tipo de expiração da ordem
   datetime                      expiration;      // Hora de expiração da ordem (para
   string                        comment;         // Comentário sobre a ordem
   ulong                         position;        // Bilhete da posição
   ulong                         position_by;     // Bilhete para uma posição oposta
  };
```

Descrição dos Campos

Campo	Descrição
action	Tipo de operação de negociação. Pode ser um dos valores da enumeração ENUM_TRADE_REQUEST_ACTIONS.
magic	Expert Advisor ID. Permite organizar um processamento analítico de ordens de negociação. Cada Expert Advisor pode definir seu próprio ID (identificador) único ao enviar uma solicitação de negociação.
order	Bilhetagem (ticket) da ordem. É usado para modificar ordens pendentes.
symbol	Ativo da ordem. Não é necessário para modificação de ordens e operações de encerramento de posição.
volume	Volume da ordem solicitada em lotes. Note que o volume real de uma operação (deal) dependerá do tipo de execução da ordem.
price	Preço no qual a ordem deve ser executada. Ordens a mercado de ativos, cujo tipo de execução é "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET), de tipo TRADE_ACTION_DEAL, não necessitam que se especifique o preço.

© 2000-2024, MetaQuotes Ltd.

Campo	Descrição
stoplimit	O valor do preço, no qual a ordem pendente Limit será colocada, quando o preço corrente alcança o valor deste *preço*(esta condição é obrigatória). Até lá, a ordem pendente não é colocada.
sl	Preço de Stop Loss no caso de um movimento desfavorável de preço
tp	Preço de Take Profit no caso de um movimento favorável de preço
deviation	O máximo desvio de preço, especificado em pontos
type	Tipo de ordem. Pode ser um dos valores da enumeração ENUM_ORDER_TYPE.
type_filling	Tipo de execução da ordem. Pode ser um dos valores da enumeração ENUM_ORDER_TYPE_FILLING.
type_time	Tipo de expiração da ordem. Pode ser um dos valores da enumeração ENUM_ORDER_TYPE_TIME.
expiration	Hora de expiração da ordem (para ordens do tipo ORDER_TIME_SPECIFIED)
comment	Comentário sobre a ordem
position	Bilhete da posição. Deve ser preenchido ao alterar e fechar a posição para a sua identificação inequívoca. Normalmente, corresponde ao bilhete de ordem, do qual, como resultado, a posição foi aberta.
position_by	Bilhete da posição oposta. Utiliza-se ao fechar uma posição usando outra oposta, se estiver aberta no mesmo instrumento, mas na direção oposta.

Para modificar ou fechar posições no sistema de cobertura, você deve especificar o seu bilhete (MqlTradeRequest::position). No sistema de compensação o bilhete também pode ser especificado, mas a identificação da posição é realizada segundo o nome do símbolo.

Para enviar ordens para realizaroperações de negociação (trade)é necessário usar a função OrderSend(). Para cada operação de negociação é necessário especificar campos obrigatórios; campos opcionais também podem ser preenchidos. Existem sete possíveis casos para enviar uma ordem de negociação:

Execução a Pedido

Esta é uma ordem de negociação para abrir uma posição no modo Request Execution (negociação sobre preços solicitados). Ela requer que se especifique os seguintes 9 campos:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

© 2000-2024, MetaQuotes Ltd.

Também é possível especificar valores dos campos "magic" e "comment".

Execução Instantânea

Esta é uma ordem de negociação para abrir uma posição no modo Instant Execution (negociação pelos preços correntes). Ela requer que se especifique os seguintes 9 campos:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Também é possível especificar valores dos campos "magic" e "comment".

Execução de Mercado

Esta é uma ordem de negociação para abrir uma posição no modo Market Execution (execução a mercado). Ela requer que se especifique os seguintes 5 campos:

- action
- symbol
- volume
- type
- type_filling

Também é possível especificar valores dos campos "magic" e "comment".

Execução Exchange

Esta é uma ordem de negociação para abrir uma posição no modo Exchange Execution (execução em um sistema de negociação externo). Ela requer que se especifique os seguintes 5 campos:

- action
- symbol
- volume
- type
- type_filling

Também é possível especificar valores dos campos "magic" e "comment".

Exemplo de operação de negociação TRADE_ACTION_DEAL para abrir uma posição Buy:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Abertura da posição Buy                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Atualização e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros do pedido
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operação de neg
   request.symbol   =Symbol();                              // símbolo
   request.volume   =0.1;                                   // volume de 0.1 lotes
   request.type     =ORDER_TYPE_BUY;                        // tipo de ordem
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // preço para abertura
   request.deviation=5;                                     // desvio permitido do pre
   request.magic    =EXPERT_MAGIC;                          // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se não for possível env
//--- informação sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Exemplo de operação de negociaçãoTRADE_ACTION_DEAL para abertura da posição Sell:

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Abertura da posição Sell                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Declaração e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros do pedido
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operação de neg
   request.symbol   =Symbol();                              // símbolo
   request.volume   =0.2;                                   // volume de 0.2 lotes
   request.type     =ORDER_TYPE_SELL;                       // tipo da ordem
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // preço para a abertura
   request.deviation=5;                                     // desvio permitido do pre
   request.magic    =EXPERT_MAGIC;                          // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // se não for possível env
//--- informação sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Exemplo de ordem de negociação TRADE_ACTION_DEAL para fechadura de posições:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Fechadura de todas as posições                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaração do pedido e o seu resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posições abertas
//--- iterar todas as posições abertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parâmetros da ordem
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- saída de informação sobre a posição
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- se o MagicNumber coincidir
      if(magic==EXPERT_MAGIC)
        {
         //--- zerar os valores do pedido e os seus resultados
         ZeroMemory(request);
         ZeroMemory(result);
         //--- configuração dos parâmetros da ordem
         request.action   =TRADE_ACTION_DEAL;        // tipo de operação de negociação
         request.position =position_ticket;          // bilhete da posição
         request.symbol   =position_symbol;          // símbolo
         request.volume   =volume;                   // volume da posição
         request.deviation=5;                        // desvio permitido do preço
         request.magic    =EXPERT_MAGIC;             // MagicNumber da posição
         //--- colocação do preço e tipo de ordem dependendo do tipo de ordem
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- saída de informação sobre o fechamento
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- envio do pedido
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se não for possível
         //--- informação sobre a operação
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

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

SL & TP Modificação

Ordem de negociação para modificar os níveis de preço StopLoss e/ou TakeProfit Ela requer que se especifique os seguintes 4 campos:

- action
- symbol
- sl
- tp
- position

Exemplo da operação de negociação TRADE_ACTION_SLTP para alterar o valor Stop-Loss e Take-Profit na posição aberta:

```
#define EXPERT_MAGIC 123456  // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Modificação do Stop-Loss e Take-Profit da posição                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declaracão do pedido e o seu resultado
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // número de posições abertas
//--- iterar sobre todas as posições abertas
   for(int i=0; i<total; i++)
     {
      //--- parâmetros da ordem
      ulong  position_ticket=PositionGetTicket(i);// bilhete da posição
      string position_symbol=PositionGetString(POSITION_SYMBOL); // símbolo
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // número d
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // MagicNumber da posição
      double volume=PositionGetDouble(POSITION_VOLUME);    // volume da posição
      double sl=PositionGetDouble(POSITION_SL);  // Stop Loss da posição
      double tp=PositionGetDouble(POSITION_TP);  // Take-Profit da posição
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- saída de informações sobre a posição
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- se o MagicNumber coincidir, o Stop-Loss e o Take-Profit não estão definido
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

```
         //--- cálculo dos níveis de preços atuais
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_I
         double price_level;
         //--- se o nível de deslocamento mínimo permitido em pontos a partir do preço
         if(stop_level<=0)
            stop_level=150; // definimos o deslocamento em 150 pontos a partir do prec
         else
            stop_level+=50; // definimos o nível de deslocamento (SYMBOL_TRADE_STOPS_I

         //--- cálculo e arredondamento dos valores Stop-Loss e Take-Profit
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(bid+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(ask-price_level,digits);
           }
         //--- zerar os valores de pedido e o seu resultado
         ZeroMemory(request);
         ZeroMemory(result);
         //--- definição dos parâmetros de operação
         request.action  =TRADE_ACTION_SLTP; // tipo de operação de negociação
         request.position=position_ticket;   // bilhete da posição
         request.symbol=position_symbol;     // símbolo
         request.sl      =sl;                // Stop Loss da posição
         request.tp      =tp;                // Take Profit da posição
         request.magic=EXPERT_MAGIC;         // MagicNumber da posição
         //--- saída de informações sobre a modificação
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- envio do pedido
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // se não for possível
         //--- informações sobre a operação
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
      }
   }
//+------------------------------------------------------------------+
```

Ordem Pendente

Ordem de negociação para colocar uma ordem pendente. Ela requer que se especifiquem os seguintes 11 campos:

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp

© 2000-2024, MetaQuotes Ltd.

- type
- type_filling
- type_time
- expiration

Também é possível especificar valores dos campos "magic" e "comment".

Exemplo de operação de negociação TRADE_ACTION_PENDING para colocação de ordens pendentes:

© 2000-2024, MetaQuotes Ltd.

```
#property description "Exemplo de colocação de ordens pendentes"
#property script_show_inputs
#define EXPERT_MAGIC 123456                              // MagicNumber do conselheiro
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // tipo da ordem
//+------------------------------------------------------------------+
//| Colocação de ordens pendentes                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- declaração e inicialização do pedido e o seu resultado
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- parâmetros para colocação da ordem pendente
   request.action   =TRADE_ACTION_PENDING;                          // tipo de oper
   request.symbol   =Symbol();                                      // símbolo
   request.volume   =0.1;                                           // volume de 0
   request.deviation=2;                                             //desvio permit
   request.magic    =EXPERT_MAGIC;                                  // MagicNumber
   int offset = 50;                                                 // deslocamento
   double price;                                                    // preço de exe
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);             // tamanho do p
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);             // número de si
   //--- verificação do tipo de operação
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                      // tipo de orde
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;    // preço para a
      request.price    =NormalizeDouble(price,digits);             // нормализовав
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                      // тип ордера
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;    // цена для от
      request.price    =NormalizeDouble(price,digits);              // preço de ab
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                           // tipo de ord
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // preço de ab
      request.price=NormalizeDouble(price,digits);                 // preço de ab
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                      // tipo de ord
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;    // preço para
      request.price    =NormalizeDouble(price,digits);             // preço de ab
     }
   else Alert("Este exemplo é apenas para colocação de ordens pendentes");   // Se a o
//--- enviar a ordem
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());            // Se não for
//--- informações sobre a operação
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Modificar Ordem Pendente

Ordem de negociação para modificar os preços de uma ordem pendente. Ela requer que se especifiquem os seguintes 7 campos:

- action
- order
- price
- sl
- tp
- type_time
- expiration

Exemplo de operação de negociação TRADE_ACTION_MODIFY para modificação dos níveis de preços da ordem pendente:

© 2000-2024, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456 // MagicNumber do conselheiro
//+------------------------------------------------------------------+
//| Modificação de ordens pendentes                                  |
//+------------------------------------------------------------------+
void OnStart()
 {
//-- declaração e inicialização do pedido e o seu resultado
 MqlTradeRequest request={};
 MqlTradeResult result={};
 int total=OrdersTotal(); // número de ordens pendentes colocadas
//--- iterar todas as ordens pendentes colocadas
 for(int i=0; i<total; i++)
 {
 //--- parâmetros da ordem
 ulong order_ticket=OrderGetTicket(i); // bilhete da ordem
 string order_symbol=Symbol(); // símbolo
 int digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // número de signos de
 ulong magic=OrderGetInteger(ORDER_MAGIC); // MagicNumber da ordem
 double volume=OrderGetDouble(ORDER_VOLUME_CURRENT); // volume atual da ordem
 double sl=OrderGetDouble(ORDER_SL); // Stop Loss atual da ordem
 double tp=OrderGetDouble(ORDER_TP); // Take Profit atual da ordem
 ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // tipo de ordem
 int offset = 50; // deslocação a partir do preço atual para colocação da ordem, em po
 double price; // preço de ativação da ordem
 double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT); // tamanho do ponto
 //--- saída de informações sobre a ordem
 PrintFormat("#%I64u %s %s %.2f %s sl: %s tp: %s [%I64d]",
 order_ticket,
 order_symbol,
 EnumToString(type),
 volume,
 DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
 DoubleToString(sl,digits),
 DoubleToString(tp,digits),
 magic);
 //--- se o MagicNumber coincidir, o Stop Loss e o Take Profit nao foram definidos
 if(magic==EXPERT_MAGIC && sl==0 && tp==0)
 {
 request.action=TRADE_ACTION_MODIFY; // tipo de operação de negociação
 request.order = OrderGetTicket(i); // bilhete da ordem
 request.symbol =Symbol(); // símbolo
 request.deviation=5; // desvio permitido a partir do preço
 //--- definição do nível de preços Take Profit e Stop Loss da ordem dependendo do seu
 if(type==ORDER_TYPE_BUY_LIMIT)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
 request.tp = NormalizeDouble(price+offset*point,digits);
 request.sl = NormalizeDouble(price-offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_SELL_LIMIT)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
 request.tp = NormalizeDouble(price-offset*point,digits);
 request.sl = NormalizeDouble(price+offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_BUY_STOP)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
 request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2024, MetaQuotes Ltd.

```
 request.sl = NormalizeDouble(price-offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 else if(type==ORDER_TYPE_SELL_STOP)
 {
 price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
 request.tp = NormalizeDouble(price-offset*point,digits);
 request.sl = NormalizeDouble(price+offset*point,digits);
 request.price =NormalizeDouble(price,digits); // preço de abertura normalizado
 }
 //--- envio do pedido
 if(!OrderSend(request,result))
 PrintFormat("OrderSend error %d",GetLastError()); // se não foi possível enviar o pe
 //--- informações sobre a operação
 PrintFormat("retcode=%u deal=%I64u order=%I64u",result.retcode,result.deal,result.or
 //--- zerado dos valores do pedido e o seu resultado
 ZeroMemory(request);
 ZeroMemory(result);
 }
 }
 }
//+------------------------------------------------------------------+
```

Excluir Ordem Pendente

Ordem de negociação para excluir uma ordem pendente. Ela requer que se especifiquem os seguintes 2 campos:

- action
- order

Exemplo de operação de negociação TRADE_ACTION_REMOVE para excluir ordens pendentes:

```
#define EXPERT_MAGIC 123456  // MagicNumber do conselheiro
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Exclusão de ordens pendentes                                     |
//+------------------------------------------------------------------+
void OnStart()
 {
//-- Declaração e inicialização do pedido e resultado
 MqlTradeRequest request={};
 MqlTradeResult result={};
 int total=OrdersTotal(); // número de ordens pendentes colocadas
//--- iterar todas as ordens pendentes colocadas
 for(int i=total-1; i>=0; i--)
 {
 ulong order_ticket=OrderGetTicket(i); // bilhete da ordem
 ulong magic=OrderGetInteger(ORDER_MAGIC); // MagicNumber da ordem
 //--- se o MagicNumber coincidir
 if(magic==EXPERT_MAGIC)
 {
 //--- zerar os valores do pedido e o resultado
 ZeroMemory(request);
 ZeroMemory(result);
 //--- instalação de parâmetros de operação
 request.action=TRADE_ACTION_REMOVE; // tipo de operação de negociação
 request.order = order_ticket; // bilhete da ordem
 //--- envio do pedido
 if(!OrderSend(request,result))
 PrintFormat("OrderSend error %d",GetLastError()); // Se não for capaz de enviar o pe
 //--- informações sobre a operação
 PrintFormat("retcode=%u deal=%I64u order=%I64u",result.retcode,result.deal,result.or
 }
 }
 }
//+------------------------------------------------------------------+
```

Também Veja

Estruturas e Classes, Funções de Negociação (Trade), Propriedades de Ordem

© 2000-2024, MetaQuotes Ltd.

A Estrutura de Resultados de uma Verificação de Solicitação de Negociação (MqlTradeCheckResult)

Antes de enviar uma solicitação de uma operação de negociação para um servidor de negociação, é recomendado verificá-la. A verificação é realizada usando a função OrderCheck(), para a qual a solicitação a ser verifica e uma variável estrutura do tipo MqlTradeCheckResult são passadas. O resultado da verificação será escrita nesta variável.

```
struct MqlTradeCheckResult
  {
   uint         retcode;             // Código da resposta
   double       balance;             // Saldo após a execução da operação (deal)
   double       equity;              // Saldo a mercado (equity) após a execução da op
   double       profit;              // Lucro flutuante
   double       margin;              // Requerimentos de Margem
   double       margin_free;         // Margem livre
   double       margin_level;        // Nível de margem
   string       comment;             // Comentário sobre o código da resposta (descric
  };
```

Descrição dos Campos

Campo	Descrição
retcode	Código de retorno
balance	Valor de saldo após a execução da operação de negociação
equity	Valor de saldo a mercado após a execução da operação de negociação
profit	Valor do lucro flutuante após a execução da operação de negociação
margin	Margem requerida para a operação de negociação
margin_free	Margem livre que sobrará após a execução da operação de negociação
margin_level	Nível de margem que será definida após a execução da operação de negociação
comment	Comentário sobre o código da resposta, descrição do erro

Também Veja

Estrutura de Solicitação de Negociação, Estrutura para Preços Correntes, OrderSend, OrderCheck

© 2000-2024, MetaQuotes Ltd.

A Estrutura de um Resultado de Solicitação de Negociação (MqlTradeResult)

Como resultado de uma solicitação de negociação, um servidor de negociação retorna dados sobre o resultado do processamento da solicitação de negociação na forma de uma estrutura predefinida especial de tipo MqlTradeResult.

```
struct MqlTradeResult
  {
   uint     retcode;          // Código de retorno da operação
   ulong    deal;             // Bilhetagem (ticket) da operação (deal),se ela for rea
   ulong    order;            // Bilhetagem (ticket) da ordem, se ela for colocada
   double   volume;           // Volume da operação (deal), confirmada pela corretora
   double   price;            // Preço da operação (deal), se confirmada pela correto
   double   bid;              // Preço de Venda corrente
   double   ask;              // Preço de Compra corrente
   string   comment;          // Comentário da corretora para a operação (por default,
   uint     request_id;       // Identificador da solicitação definida pelo terminal c
   int      retcode_external; // Código de resposta do sistema de negociação exterior
  };
```

Descrição dos Campos

Campo	Descrição
retcode	Código de retorno de um servidor de negociação
deal	Bilhetagem (ticket) da operação (deal), caso uma operação tenha sido realizada. É disponível para uma operação de negociação de tipo TRADE_ACTION_DEAL
order	Bilhetagem (ticket) da ordem, caso uma ordem tenha sido colocada. É disponível para uma operação de negociação de tipo TRADE_ACTION_PENDING
volume	Volume de operação (deal), confirmado pela corretora. Depende do tipo de preenchimento da ordem
price	Preço da operação (deal), confirmado pela corretora. Depende da área de *desvio* da requisição de negócio e/ou sobre operação de negócios
bid	O preço corrente de venda de mercado (preço re-cotado)
ask	O preço corrente de compra de mercado (preço re-cotado)
comment	O comentário da corretora para a operação (por default, ele é preenchido com a descrição código de retorno de um servidor de negociação)
request_id	Identificador da solicitação definida pelo terminal ao enviar para o servidor de negociação
retcode_external	Código de erro retornado pelo sistema de negociação exterior. O uso e tipos destes erros dependerão da corretora e do sistema de negociação

© 2000-2024, MetaQuotes Ltd.

Campo	Descrição
	exterior a partir do qual são enviadas as operações de negociação

O resultado da operação de negociação é retornado para variável do tipo MqlTradeResult, que é passado como o segundo parâmetro da função OrderSend() para realizar operações de negociação.

O terminal fixa um identificador (ID) de solicitação no campo request_id ao enviar-la para o servidor de negociação usando as funções OrdersSend() e OrderSendAsync(). O terminal recebe mensagens sobre transações realizadas provenientes do servidor de negociação e as submete para processamento através da função OnTradeTransaction() contendo os seguintes componentes como parâmetros:

- Descrição da transação de negociação na estrutura MqlTradeTransaction;
- Descrição da solicitação de negociação enviado a partir da função OrderSend() ou OrdersSendAsync(). O ID da solicitação é enviado pelo terminal para o servidor de negociação, enquanto a solicitação em si e seu request_id são armazenados na memória do terminal;
- O resultado de execução da solicitação de negociação como estrutura MqlTradeResult com o campo request_id contendo o ID desta solicitação.

A função OnTradeTransaction() recebe três parâmetros de entrada mas os últimos dois devem ser analisados somente para transações do tipo TRADE_TRANSACTION_REQUEST. Em todos os outros casos, dados sobre a solicitação de negociação e seu resultado de execução não são preenchidos. Exemplo de análise de parâmetros pode ser encontrado em Estrutura de uma Solicitação de Negociação.

A definição do request_id pelo terminal na solicitação de negociação quando enviá-la para o servidor é introduzido principalmente para trabalhar com a função assíncrona OrderSendAsync(). Este identificador permite associar a ação realizada (chamada a funções OrderSend ou OrderSendAsync) com o resultado desta ação enviada para OnTradeTransaction().

Exemplo:

```
//+------------------------------------------------------------------+
//| Enviando uma solicitação de negociação com o proc. de resultado  |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- redefine o último código de erro para zero
   ResetLastError();
//--- envia solicitação
   bool success=OrderSend(request,result);
//--- se o resultado falha - tentar descobrir o porquê
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: Requisição de negociação falhou. Erro = ",GetLastError());
      switch(answer)
        {
         //--- nova cotação
         case 10004:
           {
```

```
      Print("TRADE_RETCODE_REQUOTE");
      Print("request.price = ",request.price,"   result.ask = ",
            result.ask," result.bid = ",result.bid);
      break;
     }
    //--- ordem não é aceita pelo servidor
    case 10006:
     {
      Print("TRADE_RETCODE_REJECT");
      Print("request.price = ",request.price,"   result.ask = ",
            result.ask," result.bid = ",result.bid);
      break;
     }
    //--- preço inválido
    case 10015:
     {
      Print("TRADE_RETCODE_INVALID_PRICE");
      Print("request.price = ",request.price,"   result.ask = ",
            result.ask," result.bid = ",result.bid);
      break;
     }
    //--- SL e/ou TP inválidos
    case 10016:
     {
      Print("TRADE_RETCODE_INVALID_STOPS");
      Print("request.sl = ",request.sl," request.tp = ",request.tp);
      Print("result.ask = ",result.ask," result.bid = ",result.bid);
      break;
     }
    //--- volume inválido
    case 10014:
     {
      Print("TRADE_RETCODE_INVALID_VOLUME");
      Print("request.volume = ",request.volume,"   result.volume = ",
            result.volume);
      break;
     }
    //--- sem dinheiro suficiente para uma operação de negociação
    case 10019:
     {
      Print("TRADE_RETCODE_NO_MONEY");
      Print("request.volume = ",request.volume,"   result.volume = ",
            result.volume,"   result.comment = ",result.comment);
      break;
     }
    //--- alguma outra razão, saída com o código de resposta do servidor
    default:
     {
      Print("Outra resposta = ",answer);
```

© 2000-2024, MetaQuotes Ltd.

```
            }
         }
      //--- notifica sobre o resultado sem sucesso da solicitação de negociação retorn
      return(false);
     }
//--- OrderSend() retorna true - repete a resposta
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Estrutura de um Transação de Negociação (MqlTradeTransaction)

Ao realizar algumas ações específicas em uma conta de negociação, seu estado muda. Tais ações includem:

- Envio de uma solicitação de negociação a partir de algum aplicativo MQL5 no terminal usando as funções OrderSend e OrderSendAsync e sua posterior execução.
- Envio de uma solicitação de negociação por meio da interface gráfica do terminal e sua posterior execução.
- Ativação de ordens pendentes e ordens de stop no servidor.
- Realização de operações no lado de um servidor de negociação.

As seguintes transações de negociação são realizadas como resultado destas ações:

- tratamento de uma solicitação de negociação
- mudança de ordens de abertura
- mudança de histórico de ordens
- mudança de histórico de operações (deals)
- mudança de posições

Por exemplo, ao enviar uma ordem de mercado para compra, ele é processado, para a conta cria-se a respetiva ordem para compra, a ordem é executada, excluída da lista de ordens abertas, adicionada ao histórico de ordens, a seguir, a transação respetiva é adicionada ao histórico e, finalmente, é criada uma nova posição. Todas estas ações são transações financeiras.
Para obter transações financeiras, aplicadas à conta, a MQL5 dispõe de um processador especial chamado de OnTradeTransaction (). Ao primeiro parâmetro deste processador é transferida a estrutura MqlTradeTransaction, que descreve a transação financeira.

```
struct MqlTradeTransaction
  {
   ulong                        deal;             // Bilhetagem da operação (deal)
   ulong                        order;            // Bilhetagem da ordem
   string                       symbol;           // Nome do ativo da negociação
   ENUM_TRADE_TRANSACTION_TYPE  type;             // Tipo de transação da negociação
   ENUM_ORDER_TYPE              order_type;       // Tipo de ordem
   ENUM_ORDER_STATE             order_state;      // Estado da ordem
   ENUM_DEAL_TYPE               deal_type;        // Tipo de operação (deal)
   ENUM_ORDER_TYPE_TIME         time_type;        // Tipo de ordem por período de aç
   datetime                     time_expiration;  // Hora de expiração da ordem
   double                       price;            // Preço
   double                       price_trigger;    // Preço de ativação de ordem tipo
   double                       price_sl;         // Nível de Stop Loss
   double                       price_tp;         // Nível de Take Profit
   double                       volume;           // Volume em lotes
   ulong                        position;         // Position ticket
   ulong                        position_by;      // Ticket of an opposite position
  };
```

© 2000-2024, MetaQuotes Ltd.

Descrição dos Campos

Campo	Descrição
deal	Bilhetagem (ticket) da operação.
order	Bilhetagem (ticket) da ordem.
symbol	O nome do ativo da negociação, para o qual a transação é realizada.
type	Tipo de transação de negociação. O valor pode ser um dos valores da enumeração ENUM_TRADE_TRANSACTION_TYPE.
order_type	Tipo de ordem de negociação. O valor pode ser um dos valores da enumeração ENUM_ORDER_TYPE.
order_state	Estado da ordem de negociação. O valor pode ser um dos valores da enumeração ENUM_ORDER_STATE.
deal_type	Tipo de operação. O valor pode ser um dos valores da enumeração ENUM_DEAL_TYPE.
time_type	Tipo de ordem quanto a expiração. O valor pode ser um dos valores da enumeração ENUM_ORDER_TYPE_TIME.
time_expiration	Período de expiração de uma ordem pendente (para ordens dos tipos ORDER_TIME_SPECIFIED e ORDER_TIME_SPECIFIED_DAY).
price	Preço. Dependendo do tipo de transação de negociação, ele pode ser um preço de uma ordem, de uma operação (deal), ou de uma posição.
price_trigger	Preço stop (de ativação) de uma ordem Stop Limit (ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Preço Stop Loss. Dependendo do tipo de transação de negociação, ele pode estar relacionado a uma ordem, a uma operação (deal) ou a uma posição.
price_tp	Preço Take Profit. Dependendo do tipo de transação de negociação, ele pode estar relacionado a uma ordem, a uma operação (deal) ou a uma posição.
volume	Volume em lotes. Dependendo do tipo de transação de negociação, ele pode indicar o valor corrente de uma ordem, de uma operação (deal), ou de uma posição.
position	Bilhete da posição que foi afetado pela transação.
position_by	Bilhete da posição oposta. Utiliza-se ao fechar uma posição usando outra oposta, se estiver aberta no mesmo instrumento, mas na direção oposta.

O parâmetro essencial para análise de uma transação recebida é o seu tipo, especificado no campo **type**. Por exemplo, se uma transação é do tipo TRADE_TRANSACTION_REQUEST (um resultado de operar uma negociação requisitada pelo servidor foi recebido), a estrutura possui apenas um campo único que é completamente preenchido - **type**. Outros campos não são analisados. Neste caso, nós podemos analisar dois parâmetros adicionais, **request** e **result**, submetidos ao handler OnTradeTransaction(), como mostrado abaixo.

© 2000-2024, MetaQuotes Ltd.

Tendo dados sobre um tipo de operação de negociação, você pode decidir sobre a análise do estado corrente de ordens, posições e operações (deals) em um conta de negociação. Lembre-se que uma solicitação de negociação enviada ao servidor a partir do terminal pode gerar várias novas transações. A prioridade de chegada delas no terminal não é garantida.

A estrutura MqlTradeTransaction é preenchida de diferentes maneiras dependendo do tipo de transação de negociação (ENUM_TRADE_TRANSACTION_TYPE):

TRADE_TRANSACTION_ORDER_* e TRADE_TRANSACTION_HISTORY_*

Os seguintes campos na estrutura MqlTradeTransaction são preenchidos para transações de negociação relacionados a manipulação de abertura de ordens (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE e TRADE_TRANSACTION_ORDER_DELETE) e histórico de ordens (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE):

- order - bilhetagem da ordem;
- symbol - nome do ativo da ordem;
- type - tipo de transação de negociação;
- order_type - tipo de ordem;
- orders_state - estado corrente da ordem;
- time_type - tipo de expiração da ordem;
- time_expiration - hora de expiração da ordem (para ordens tendo tipos de expiração ORDER_TIME_SPECIFIED e ORDER_TIME_SPECIFIED_DAY);
- price - preço da ordem especificado por um cliente;
- price_trigger - preço de stop de ordem stop limit (somente para ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT);
- price_sl - Preço de ordem Stop Loss (preenchido, se especificado na ordem);
- price_tp - Preço de ordem Take Profit (preenchido, se especificado na ordem);
- volume - volume corrente da ordem (não preenchida). Volume inicial da ordem pode ser encontra no histórico de ordens usando a função HistoryOrders*.
- position - bilhete da posição, aberta, fechada ou modificada como resultado da execução da ordem. A preenchido apenas para ordens de mercado. Não é preenchido para TRADE_TRANSACTION_ORDER_ADD.
- position_by - bilhete da posição oposta. Preenche-se apenas para ordens de fechamento da posição oposta (close by).

TRADE_TRANSACTION_DEAL_*

Os seguintes campos na estrutura MqlTradeTransaction são preenchidos para transações de negociação relacionados a manipulação de operações (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE e TRADE_TRANSACTION_DEAL_DELETE):

- deal - bilhetagem da operação (deal);
- order - bilhetagem da ordem, baseado na qual uma operação (deal) foi realizada;
- symbol - nome de ativo da operação (deal);
- type - tipo de transação de negociação;
- deal_type - tipo de operação (deal);
- price - preço de operação (deal);
- price_sl - preço de Stop Loss (preenchido, se especificado na ordem, baseado na qual uma operação foi realizada);

© 2000-2024, MetaQuotes Ltd.

- price_tp - preço de Take Profit (preenchido, se especificado na ordem, baseado na qual uma operação foi realizada);
- volume - volume de operação em lotes.
- position - bilhete da posição, aberta, fechada ou modificada como resultado da execução da transação.
- position_by - bilhete da posição oposta. Preenche-se apenas para transações de fechamento da posição oposta (out by).

TRADE_TRANSACTION_POSITION

Os seguintes campos na estrutura MqlTradeTransaction são preenchidos para transações de negociação relacionado a mudança de posições não conectados com execução de operações (deals) (TRADE_TRANSACTION_POSITION):

- symbol - nome de ativo da posição;
- type - tipo de transação de negociação;
- deal_type - tipo de posição (DEAL_TYPE_BUY ou DEAL_TYPE_SELL);
- price - preço médio ponderado de abertura de posição;
- price_sl - preço de Stop Loss;
- price_tp - preço de Take Profit;
- volume - volume de posição em lotes, se ele foi alterado.
- position - bilhete de posição.

Alteração de Posição (adição, alteração, encerramento), como resultado de uma execução de operação (deal), não leva a ocorrência de transação de TRADE_TRANSACTION_POSITION.

TRADE_TRANSACTION_REQUEST

Somente um campo na estrutura MqlTradeTransaction é preenchido para transações de negociação descrevendo o fato de que uma solicitação de negociação foi processada por um servidor e que o resultado do processamento foi recebido (TRADE_TRANSACTION_REQUEST):

- type - tipo de transação de negociação;

Somente o campo type (tipo de transação de negociação) de ser analisado para tais transações. O segundo e terceiro parâmetros (request e result) da função OnTradeTransaction deve ser analisado para dados adicionais.

Exemplo:

```
input int MagicNumber=1234567;

//--- habilita a classe de negociação CTrade e declara a variável desta classe
#include <Trade\Trade.mqh>
CTrade trade;
//--- flags para instalação e exclusão de ordens pendentes
bool pending_done=false;
bool pending_deleted=false;
//--- bilhetagem da ordem pendente será armazenada aqui
ulong order_ticket;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//--- define o MagicNumber para marcar todas as nossas ordens
   trade.SetExpertMagicNumber(MagicNumber);
//--- solicitações de negociação serão enviadas em modo assíncrono usando a função Orc
   trade.SetAsyncMode(true);
//--- inicializa a variável em zero
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---instalando uma ordem pendente
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
      //--- caso a função BuyStop() executou com sucesso
      if(res)
        {
         pending_done=true;
         //--- obtém um resultado da solicitação enviando para CTrade
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- obtém o request_id para a solicitações enviada
         uint request_id=trade_result.request_id;
         Print("Pedido foi enviado para definir uma ordem pendente. Request_ID=",reque
         //--- armazena a bilhetagem da ordem (será zero se usar o modo assíncrono de
         order_ticket=trade_result.order;
         //--- tudo está feito, saí cedo do handler OnTick()
         return;
        }
     }
//--- exclui a ordem pendente
   if(!pending_deleted)
      //--- verificação adicional
      if(pending_done && (order_ticket!=0))
        {
         //--- tenda excluir a ordem pendente
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- quando solicitação de exclusão é enviada com sucesso
         if(res)
           {
```

```
            pending_deleted=true;
            //--- obtém o resultado da execução da solicitação
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- peque o ID da solicitação proveniente do resultado
            uint request_id=trade_result.request_id;
            //--- exibe no Diário
            Print("O pedido foi enviado para eliminar uma ordem pendente #",order_tic
                  ". Request_ID=",request_id,
                  "\r\n");
            //--- fixa a bilhetagem da ordem proveniente do resultado da solicitação
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------+
//| Função TradeTransaction                                          |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- obtém o tipo de transação como valor de enumeração
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- se a transação é a solicitação de manipulação do resultado, somente seu nome é e
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- exibe o a string do nome da solicitação manipulada
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- exibe a descrição do resultado da solicitação
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- armazena a bilhetagem da ordem para sua exclusão na próxima manipulação em
      if(result.order!=0)
        {
         //--- exclui esta ordem através de sua bilhetagem na próxima chamada de OnTic
         order_ticket=result.order;
         Print(" Bilhetagem da ordem pendente ",order_ticket,"\r\n");
        }
     }
   else // exibe a descrição completa para transações de um outro tipo
//--- exibe a descriçaõ da transação recebida no Diário
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| Retorna a descrição textual da transação                         |
```

```
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Ativo: "+trans.symbol+"\r\n";
   desc+="Bilhetagem (ticket) da operação: "+(string)trans.deal+"\r\n";
   desc+="Tipo de operação: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Bilhetagem (ticket) da ordem: "+(string)trans.order+"\r\n";
   desc+="Tipo de ordem: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Estado da ordem: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Ordem do tipo time: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Expiração da ordem: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Preço: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Gatilho do preço: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
//--- retorna a string obtida
   return desc;
  }
//+------------------------------------------------------------------+
//| Retorna a descrição textual da solicitação de negociação         |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Ativo: "+request.symbol+"\r\n";
   desc+="Número mágico: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Bilhetagem (ticket) da ordem: "+(string)request.order+"\r\n";
   desc+="Tipo de ordem: "+EnumToString(request.type)+"\r\n";
   desc+="Preenchimento da ordem: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Ordem do tipo time: "+EnumToString(request.type_time)+"\r\n";
   desc+="Expiração da ordem: "+TimeToString(request.expiration)+"\r\n";
   desc+="Preço: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Pontos de desvio: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Comentário: "+request.comment+"\r\n";
//--- retorna a string obtida
   return desc;
  }
//+------------------------------------------------------------------+
//| Retorna a desc. textual do resultado da manipulação da solic.    |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="ID da solicitação: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Bilhetagem (ticket) da ordem: "+(string)result.order+"\r\n";
   desc+="Bilhetagem (ticket) da operação: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Preço: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Compra: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Venda: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comentário: "+result.comment+"\r\n";
//--- retorna a string obtida
   return desc;
  }
```

Também Veja

Tipos de Transação de Negociação, OnTradeTransaction()

© 2000-2024, MetaQuotes Ltd.

A Estrutura para Retornar Preços Correntes (MqlTick)

Esta é uma estrutura para armazenar os últimos preços do ativo. Ela é projetada para recuperação rápida da maioria das informações solicitadas sobre preços correntes.

```
struct MqlTick
  {
   datetime     time;          // Hora da última atualização de preços
   double       bid;           // Preço corrente de venda
   double       ask;           // Preço corrente de compra
   double       last;          // Preço da última operação (preço último)
   ulong        volume;        // Volume para o preço último corrente
   long         time_msc;      // Tempo do "Last" preço atualizado em  milissegundos
   uint         flags;         // Flags de tick
   double       volume_real;   // Volume para o preço Last atual com maior precisão
  };
```

A variável do tipo MqlTick permite obter valores de Preço de Compra, Preço de Venda, Preço Último e Volume com uma única chamada da função SymbolInfoTick().

Os parâmetros de cada tick são preenchidos independentemente se existem alterações comparadas com o tick anterior. Assim, é possível descobrir o preço correto para qualquer momento no passado, sem a necessidade de procurar os valores anteriores do histórico do tick. Por exemplo, mesmo que apenas altere o preço Bid durante a chegada do tick, a estrutura ainda contém outros parâmetros, bem como incluindo o preço Ask anterior, volume, etc.

Você pode analisar as flags dos ticks para descobrir quais dados foram alterados exatamente:

- TICK_FLAG_BID - tick alterou o preço Bid
- TICK_FLAG_ASK - tick alterou o preço Ask
- TICK_FLAG_LAST - tick alterou o último preço da oferta
- TICK_FLAG_VOLUME - tick alterou o volume
- TICK_FLAG_BUY - tick é resultado de uma compra
- TICK_FLAG_SELL - tick é resultado de uma venda

Exemplo:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() falhou, erro = ",GetLastError());
//---
  }
```

Também Veja

© 2000-2024, MetaQuotes Ltd.

Estruturas e Classes, CopyTicks(), SymbolInfoTick()

© 2000-2024, MetaQuotes Ltd.

Esta seção descreve as estruturas para trabalhar com o Calendário econômico que está disponível diretamente na plataforma MetaTrader. O calendário econômico é uma enciclopédia pronta com descrição, data de lançamento e grau de importância de indicadores macroeconômicos. Os valores reais dos indicadores macroeconômicos entram na plataforma MetaTrader imediatamente no momento da publicação e são exibidos no gráfico como tags, permitindo monitorar visualmente os indicadores necessários por países, por moedas e por importância.

As funções do calendário econômico permitem analisar automaticamente os eventos recebidos de acordo com critérios de importância individuais - países/pares de moedas.

As descrições do país são especificadas pela estrutura MqlCalendarCountry. São usadas as funções CalendarCountryById() e CalendarCountries()

```
struct MqlCalendarCountry
  {
   ulong                                id;                    // identificador de país
   string                               name;                  // nome do país (na cod
   string                               code;                  // nome de código do pa
   string                               currency;              // código da moeda do p
   string                               currency_symbol;       // símbolo da moeda do
   string                               url_name;              // nome do país usado n
  };
```

As descrições dos eventos são especificadas pela estrutura MqlCalendarEvent. É usado nas funções CalendarEventById(), CalendarEventByCountry() e CalendarEventByCurrency()

```
struct MqlCalendarEvent
  {
   ulong                                id;                    // identificador de eve
   ENUM_CALENDAR_EVENT_TYPE             type;                  // tipo de evento da en
   ENUM_CALENDAR_EVENT_SECTOR           sector;                // setor ao qual está r
   ENUM_CALENDAR_EVENT_FREQUENCY        frequency;             // frequência do evento
   ENUM_CALENDAR_EVENT_TIMEMODE         time_mode;             // modo de hora do even
   ulong                                country_id;            // identificador do paí
   ENUM_CALENDAR_EVENT_UNIT             unit;                  // unidade de medida da
   ENUM_CALENDAR_EVENT_IMPORTANCE       importance;            // importância do event
   ENUM_CALENDAR_EVENT_MULTIPLIER       multiplier;            // multiplicador da lei
   uint                                 digits;                // número de casas deci
   string                               source_url;            // URL da fonte em que
   string                               event_code;            // código do evento
   string                               name;                  // nome do evento na li
  };
```

Os valores do evento são definidos pela estrutura MqlCalendarValue. É usado nas funções CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() e CalendarValueLast()

© 2000-2024, MetaQuotes Ltd.

```
struct MqlCalendarValue
  {
   ulong                                id;                    // ID do valor
   ulong                                event_id;              // ID do evento
   datetime                             time;                  // hora e data do event
   datetime                             period;                // período de relatório
   int                                  revision;              // revisão do indicador
   long                                 actual_value;          // valor real em ppm ou
   long                                 prev_value;            // valor anterior em pp
   long                                 revised_prev_value;    // valor anterior revis
   long                                 forecast_value;        // valor previsto em pp
   ENUM_CALENDAR_EVENT_IMPACT           impact_type;           // impacto potencial na
  //--- funções para verificar valores
   bool                          HasActualValue(void) const;   // retorna true se no c
   bool                          HasPreviousValue(void) const; // retorna true se no c
   bool                          HasRevisedValue(void) const;  // retorna true se no c
   bool                          HasForecastValue(void) const; // retorna true se no c
  //--- funções para obter valores
   double                        GetActualValue(void) const;   // retorna actual_value
   double                        GetPreviousValue(void) const; // retorna prev_value o
   double                        GetRevisedValue(void) const;  // retorna revised_prev_
   double                        GetForecastValue(void) const; // retorna forecast_val
  };
```

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos *actual_value*, *forecast_value*, *prev_value* e *revised_prev_value*. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Exemplo de processamento de eventos de calendário:

```
//--- criamos nossa própria estrutura para armazenar eventos de calendário com valores
struct AdjustedCalendarValue
  {
   ulong                                id;                    // ID do valor
   ulong                                event_id;              // ID do evento
   datetime                             time;                  // hora e data do event
   datetime                             period;                // período de relatório
   int                                  revision;              // revisão do indicador
   double                               actual_value;          // valor atual do indic
   double                               prev_value;            // valor anterior do in
   double                               revised_prev_value;    // valor anterior revis
```

```
   double                              forecast_value;        // valor previsto do inc
   ENUM_CALENDAR_EVENT_IMPACT          impact_type;           // impacto potencial na
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- código do país para a União Europeia de acordo com o padrão ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtemos todos os valores dos eventos para a União Europeia
   MqlCalendarValue values[];
//--- definimos os limites do intervalo do qual tomamos os eventos
   datetime date_from=D'01.01.2021';  // tomamos todos os eventos de 2021
   datetime date_to=0;                // 0 representa todos os eventos conhecidos, mes
//--- solicitamos o histórico do evento para a União Europeia desde 2021
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Erro! Falha ao obter eventos por país country_code=%s", EU_code);
      PrintFormat("Código de erro: %d", GetLastError());
      return;
     }
   else
      PrintFormat("Recebidos valores de eventos para o país country_code=%s: %d",
                  EU_code, ArraySize(values));
//--- reduzimos o tamanho da matriz para exibição no Diário
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- exibimos os valores dos eventos no Diário tal qual como estão, sem verificações
   Print("Exibimos os valores do calendário tal qual como estão");
   ArrayPrint(values);

//--- verificamos os valores dos campos e os atualizamos para os valores atuais
//--- primeira opção para verificar e obter valores
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- copiarmos os valores verificados e corrigidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- efetuamos a verificação de valores e dividimos por 1000 000
      if(values[i].actual_value==LONG_MIN)
```

```
         values_adjusted_1[i].actual_value=double("nan");
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1[i].revised_prev_value=values[i].revised_prev_value/1000000.

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("Primeira maneira de verificar e obter valores do calendário");
   ArrayPrint(values_adjusted_1);

//--- segunda opção para verificar e obter valores
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- copiarmos os valores verificados e corrigidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- efetuamos as verificações e obtemos valores
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");
```

© 2000-2024, MetaQuotes Ltd.

```
      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("Segunda maneira de verificar e obter valores do calendário");
   ArrayPrint(values_adjusted_2);

//--- terceira opção para verificar e obter valores
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- copiarmos os valores verificados e corrigidos
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- obtemos valores sem verificações
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("Terceira maneira de obter valores do calendário, sem verificações");
   ArrayPrint(values_adjusted_3);
  }
/*
   Recebidos os valores de eventos segundo country_code=EU: 1051
  Exibimos os valores do calendário tal qual como estão
         [id] [event_id]              [time]            [period] [revision]       [act
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -922337203
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
  Primeira maneira de verificar e obter valores de calendário
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  Segunda maneira de verificar e obter valores do calendário
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
```

© 2000-2024, MetaQuotes Ltd.

```
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  Terceira maneira de obter valores do calendário, sem verificações
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
*/
```

A frequência do evento é indicada na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_FREQUENCY**

Identificador	Descrição
CALENDAR_FREQUENCY_NONE	Frequência de publicação não definida
CALENDAR_FREQUENCY_WEEK	Publicação uma vez por semana
CALENDAR_FREQUENCY_MONTH	Publicação uma vez por mês
CALENDAR_FREQUENCY_QUARTER	Publicação uma vez por trimestre
CALENDAR_FREQUENCY_YEAR	Publicação uma vez por ano
CALENDAR_FREQUENCY_DAY	Publicação uma vez por dia

O tipo de evento é indicado na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_TYPE**

Identificador	Descrição
CALENDAR_TYPE_EVENT	Evento (reunião, discurso e assim por diante)
CALENDAR_TYPE_INDICATOR	Indicador
CALENDAR_TYPE_HOLIDAY	Feriado

O setor econômico do evento é indicado na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_SECTOR**

Identificador	Descrição
CALENDAR_SECTOR_NONE	Setor não definido
CALENDAR_SECTOR_MARKET	Mercado, bolsa

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
CALENDAR_SECTOR_GDP	Produto Interno Bruto (PIB)
CALENDAR_SECTOR_JOBS	Mercado laboral
CALENDAR_SECTOR_PRICES	Preços
CALENDAR_SECTOR_MONEY	Dinheiro
CALENDAR_SECTOR_TRADE	Comércio
CALENDAR_SECTOR_GOVERNMENT	Governo
CALENDAR_SECTOR_BUSINESS	Negócio
CALENDAR_SECTOR_CONSUMER	Consumo
CALENDAR_SECTOR_HOUSING	Alojamento
CALENDAR_SECTOR_TAXES	Impostos
CALENDAR_SECTOR_HOLIDAYS	Feriados

A importância do evento é indicada na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_IMPORTANCE**

Identificador	Descrição
CALENDAR_IMPORTANCE_NONE	Nível de importância não definido
CALENDAR_IMPORTANCE_LOW	Baixa importância
CALENDAR_IMPORTANCE_MODERATE	Importância média
CALENDAR_IMPORTANCE_HIGH	Alta importância

O tipo de unidade das leituras do evento é indicado na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_UNIT**

Identificador	Descrição
CALENDAR_UNIT_NONE	Unidade não definida
CALENDAR_UNIT_PERCENT	Juros
CALENDAR_UNIT_CURRENCY	Moeda nacional
CALENDAR_UNIT_HOUR	Horas
CALENDAR_UNIT_JOB	Número de empregos
CALENDAR_UNIT_RIG	Equipamentos de perfuração
CALENDAR_UNIT_USD	Dólares estadunidenses

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
CALENDAR_UNIT_PEOPLE	Número de pessoas
CALENDAR_UNIT_MORTGAGE	Número de empréstimos hipotecários
CALENDAR_UNIT_VOTE	Número de votos
CALENDAR_UNIT_BARREL	Número de barris
CALENDAR_UNIT_CUBICFEET	Volume em pés cúbicos
CALENDAR_UNIT_POSITION	Posições líquidas de especuladores
CALENDAR_UNIT_BUILDING	Número de edifícios

Em alguns casos, os valores de um indicador econômico exigem a indicação de um multiplicador que é indicado na estrutura MqlCalendarEvent. Possíveis valores de multiplicadores são listados **ENUM_CALENDAR_EVENT_MULTIPLIER**

Identificador	Descrição
CALENDAR_MULTIPLIER_NONE	Multiplicador não definido
CALENDAR_MULTIPLIER_THOUSANDS	Milhares
CALENDAR_MULTIPLIER_MILLIONS	Milhões
CALENDAR_MULTIPLIER_BILLIONS	Bilhões
CALENDAR_MULTIPLIER_TRILLIONS	Trilhões

O impacto potencial do evento na taxa de câmbio nacional é indicado na estrutura MqlCalendarValue. Possíveis valores são listados **ENUM_CALENDAR_EVENT_IMPACT**

Identificador	Descrição
CALENDAR_IMPACT_NA	Impacto não especificado
CALENDAR_IMPACT_POSITIVE	Impacto positivo
CALENDAR_IMPACT_NEGATIVE	Impacto negativo

A hora do evento é indicada na estrutura MqlCalendarEvent. Possíveis valores são listados **ENUM_CALENDAR_EVENT_TIMEMODE**

Identificador	Descrição
CALENDAR_TIMEMODE_DATETIME	A fonte publica a hora exata do evento
CALENDAR_TIMEMODE_DATE	Evento leva o dia todo
CALENDAR_TIMEMODE_NOTIME	A fonte não publica a hora do evento

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
CALENDAR_TIMEMODE_TENTATIVE	A fonte não publica previamente a hora exata do evento, apenas seu dia. A hora é especificada quando o evento ocorre.

Veja também

Calendário econômico

© 2000-2024, MetaQuotes Ltd.

Códigos de Erros e Avisos

Esta seção contém as seguintes descrições:

- Códigos de retorno do servidor de negociação – analisando resultados da solicitação de negociação enviada pela função OrderSend();
- Avisos do compilador – códigos de mensagens de aviso que aparecem na compilação (não erros);
- Erros de compilação – códigos de mensagens de erro em uma tentativa sem sucesso de compilar;
- Erros em tempo de execução – códigos de erro na execução de programas-mql5, que podem ser obtidos usando a função GetLastErtror().

© 2000-2024, MetaQuotes Ltd.

Códigos de Retorno do Servidor de Negociação

Todas as solicitações para executar operações de negociação são enviadas como uma estrutura de uma solicitação de negociaçãoMqlTradeRequestusando a funçãoOrderSend(). O resultado da execução da função é colocado na estrutura MqlTradeResult, cujo campo *retcode* contém o código de retorno do servidor de negociação.

Código	Constante	Descrição
10004	TRADE_RETCODE_REQUOTE	Nova Cotação
10006	TRADE_RETCODE_REJECT	Solicitação rejeitada
10007	TRADE_RETCODE_CANCEL	Solicitação cancelada pelo negociador
10008	TRADE_RETCODE_PLACED	Ordem colocada
10009	TRADE_RETCODE_DONE	Solicitação concluída
10010	TRADE_RETCODE_DONE_PARTIAL	Somente parte da solicitação foi concluída
10011	TRADE_RETCODE_ERROR	Erro de processamento de solicitação
10012	TRADE_RETCODE_TIMEOUT	Solicitação cancelada por expiração de tempo (timeout)
10013	TRADE_RETCODE_INVALID	Solicitação inválida
10014	TRADE_RETCODE_INVALID_VOLUME	Volume inválido na solicitação
10015	TRADE_RETCODE_INVALID_PRICE	Preço inválido na solicitação
10016	TRADE_RETCODE_INVALID_STOPS	Stops inválidos na solicitação
10017	TRADE_RETCODE_TRADE_DISABLED	Negociação está desabilitada
10018	TRADE_RETCODE_MARKET_CLOSED	Mercado está fechado
10019	TRADE_RETCODE_NO_MONEY	Não existe dinheiro suficiente para completar a solicitação
10020	TRADE_RETCODE_PRICE_CHANGED	Os preços se alteraram
10021	TRADE_RETCODE_PRICE_OFF	Não existem cotações para processar a solicitação
10022	TRADE_RETCODE_INVALID_EXPIRATION	Data de expiração da ordem inválida na solicitação
10023	TRADE_RETCODE_ORDER_CHANGED	Estado da ordem alterada
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	Solicitações com freqüência em excesso
10025	TRADE_RETCODE_NO_CHANGES	Sem alterações na solicitação
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading desabilitado pelo servidor
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading desabilitado pelo terminal cliente

© 2000-2024, MetaQuotes Ltd.

Código	Constante	Descrição
10028	TRADE_RETCODE_LOCKED	Solicitação bloqueada para processamento
10029	TRADE_RETCODE_FROZEN	Ordem ou posição congeladas
10030	TRADE_RETCODE_INVALID_FILL	Tipo de preenchimento de ordem inválido
10031	TRADE_RETCODE_CONNECTION	Sem conexão com o servidor de negociação
10032	TRADE_RETCODE_ONLY_REAL	Operação é permitida somente para contas reais
10033	TRADE_RETCODE_LIMIT_ORDERS	O número de ordens pendentes atingiu o limite
10034	TRADE_RETCODE_LIMIT_VOLUME	O volume de ordens e posições para o ativo atingiu o limite
10035	TRADE_RETCODE_INVALID_ORDER	Incorreto ou proibida tipo de ordem
10036	TRADE_RETCODE_POSITION_CLOSED	Posição com a especificação POSITION_IDENTIFIER já foi fechado
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	O volume que se pode fechar excede o volume da posição atual
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Para a posição indicada já existe uma ordem de fechamento. Pode aoontecer no trabalho no sistema de cobertura: • ao tentar fechar a posição oposta, se já existir uma ordem para fechamento desta posição • ao tentar fechar completa ou parcialmente, se o volume total de ordens existentes para fechamento e o volume total da ordem re-colocada excede o volume da posição atual
10040	TRADE_RETCODE_LIMIT_POSITIONS	O número de posições abertas, que pode estar simultaneamente na conta, pode ser limitado pelas definições do servidor. Quando é atingido o limite, em resposta à colocação de uma ordem, o servidor retorna o erro TRADE_RETCODE_LIMIT_POSITIONS. A restrição opera de forma diferente dependendo do tipo de cálculo para as posições da conta: • Sistema de compensação — considera o número de posições abertas. Ao atingir o limite, a plataforma não permite colocar novas ordens, cuja execução

© 2000-2024, MetaQuotes Ltd.

Código	Constante	Descrição
		pode aumentar o número de posições abertas. Na verdade, a plataforma irá colocar ordens apenas para os símbolos que já têm posições abertas. No sistema de compensação, durante a verificação do limite, não são levadas em conta as ordens pendentes atuais, uma vez que sua execução pode conduzir a uma alteração nas posições atuais, em vez de um aumento do seu número. • Sistema de cobertura — além das posições abertas, são consideradas as ordens pendentes colocadas, uma vez que sua execução sempre conduz à apertura de uma nova posição. Ao atingir o limite, a plataforma não permite colocar tanto ordens de mercado para abrir posições quanto ordens pendentes.
10041	TRADE_RETCODE_REJECT_CANCEL	O pedido de ativação da ordem pendente foi rejeitado, e a ordem cancelada
10042	TRADE_RETCODE_LONG_ONLY	O pedido foi rejeitado, porque, no símbolo, foi definida a regra "**Somente são permitidas posições longas**" (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	O pedido foi rejeitado, porque, no símbolo, foi definida a regra "**Somente são permitidas posições curtas**" (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	O pedido foi rejeitado, porque, no símbolo, foi definida a regra "**Só é permitido o fechamento de posição**"
10045	TRADE_RETCODE_FIFO_CLOSE	O pedido foi rejeitado, porque, no símbolo, foi definida a regra "**Só é permitido o fechamento da posição existente segundo o princípio FIFO**" (ACCOUNT_FIFO_CLOSE=true)
10046	TRADE_RETCODE_HEDGE_PROHIBITED	O pedido foi rejeitado, porque, no símbolo, foi definida a regra "**Proibido abrir posições opostas segundo um mesmo símbolo**". Por exemplo, se a conta tiver uma posição Buy, o usuário não poderá abrir uma posição Sell ou colocar uma ordem de venda pendente. A regra só pode ser aplicada em contas com

© 2000-2024, MetaQuotes Ltd.

Código	Constante	Descrição
		sistema de cobertura (ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

© 2000-2024, MetaQuotes Ltd.

Avisos do Compilador

Os avisos do compilador são exibidos com propósitos informativos somente e não são mensagens de erro.

Código	Descrição
21	Registro incompleto de uma data na string datetime.
22	Número errado na string datetime para a data. Requerimentos: Ano 1970 <= X <= 3000 Mês 0 <X <= 12 Dia 0 <X <= 31/30/28 (29)….
23	Número errado na string datetime para hora. Requerimentos: Hora 0 <= X <24 Minuto 0 <= X <60
24	Cor inválida no formato RGB: um dos componentes RGB é menor que 0 ou maior que 255
25	Caractere desconhecido das seqüências de escape. Conhecido: \n \r \t \\ \" \' \X \x
26	Volume grande demais de variáveis locais (> 512Kb) da função, reduza o número
29	Enumeração já definida (duplicação) - membros serão adicionados à primeira definição
30	Sobreposição (overriding) de macro
31	A variável está declarada mas não é usada em nenhum lugar
32	Construtor deve ser do tipo void
33	Destrutor deve ser do tipo void
34	Constante não se encaixa na faixa de inteiros (X > _UI64_MAX || X < _I64_MIN) e será convertido para o tipo double
35	HEX longo demais - mais que 16 caracteres significativos (grupos de 4 bits de posições mais elevadas são cortados)
36	Sem grupos de 4 bits na string HEX "0x"
37	Sem função - nada a ser executado
38	Uma variável não inicializada sendo usada
41	Função não tem corpo, e não é chamada
43	Possível perda de dados na conversão Exemplo: int x = (double) z;
44	Perda de precisão (de dados) durante conversão de uma constante. Exemplo: int x = M_PI
45	Diferença entre os sinais de operandos em operações de comparação. Exemplo: (char) c1> (uchar) c2

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
46	Problemas com importação de função - declaração de #import é requerida ou o import de funções está fechado
47	Descrição grande demais - caracteres extras não serão incluídos no arquivo executável
48	O número de buffers de indicador declarados é menor que o requerido
49	Sem cor para plotar um série gráfica no indicador
50	Sem série gráfica para desenhar o indicador
51	Função Handler 'OnStart' não encontrado no script
52	Função handler 'OnStart' está definido com parâmetros errados
53	Função 'OnStart' pode ser definida somente em um script
54	Função 'OnInit' está definida com parâmetros errados
55	Função 'OnInit' não é usada em scripts
56	Função 'OnDeinit' está definida com parâmetros errados
57	Função 'OnDeinit' não é usada em scripts
58	Duas funções 'OnCalculate' estão definidas. OnCalculate() com um array de preçoserá usada
59	Sobre-preenchimento detectado ao calcular uma constante inteira complexa
60	Provavelmente, a variável não está inicializada.
61	Este declaração torna impossível de referenciar a variável local declarada na linha especificada
62	Esta declaração torna impossível de referenciar a variável global declarada na linha especificada
63	Não pode ser usada para array alocado estaticamente
64	Esta declaração de variável oculta variável predefinida
65	O valor da expressão é sempre true/false
66	Usar um variável ou expressão de tipo bool em operações matemáticas não é seguro
67	O resultado da aplicação de operador menos unário para um tipo sem sinal ulong é indefinido
68	A versão especificada na propriedade #property version não é aceitável para a seção Market;o formato correto de #property version é "XXX.YYY"
69	Declaração controlada vazia encontrada
70	Tipo de retorno de função inválido ou parâmetros incorretos durante declaração da função de manipulação de evento

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
71	Uma implícita conversão de estruturas para um tipo é requerida
72	Esta declaração torna direto acesso ao membro de uma classe declarada na string especificada impossível. Acesso somente será possível com o operador de resolução de escopo ::
73	A constante binária é muito grande, os dígitos de alta ordem serão truncados
74	O parâmetro no método da classe herança tem um diferente modificador const, a função derivada tem sobrecarregado a função geradora
75	Negativo ou deslocamento com valor muito grande em deslocamento de operação bit a bit, resultado da execução é indefinido
76	Função deve retornar um valor
77	função void retorna um valor
78	Nem todos os caminhos de controle retornam um valor
79	Expressões não são permitidas em uma escala global
80	Verifica a precedência do operador para um possível erro; usa parênteses para esclarecer a precedência
81	Foi definido duas OnCalCulate(). Será utilizado o modo OHLC
82	Struct sem membros, atribuído o tamanho de 1 byte
83	Valor de retorno da função deve ser verificado
84	Recursos do indicador é compilado para depuração. Isso retarda o desempenho. Por favor, recompile o indicador para aumentar o desempenho
85	Código de caracteres muito grande na linha, deve estar na faixa de 0 a 65535
86	Caractere não reconhecido na linha
87	Nenhuma propriedade da janela do indicador (definindo a exibição na janela principal ou na sub-janela) foi definida. Propriedade #property indicator_chart_window foi aplicada

© 2000-2024, MetaQuotes Ltd.

Erros de Compilação

O MetaEditor 5 exibe mensagens de erro sobre os erros de programa detectados pelo compilador interno durante a compilação. A lista destes erros estão na tabela dada abaixo. Para compilar um código-fonte em um código executável, pressione **F7**. Programas que contêm erros não podem ser compilados até que os erros identificados pelo compilador sejam eliminados.

Código	Descrição
100	Erro de leitura de arquivo
101	Erro de abertura de arquivo *. EX5 para escrever nele para gravação
103	Sem memória livre suficiente para completar compilação
104	Unidade sintática vazia não reconhecida pelo compilador
105	Nome de arquivo incorreto em #include
106	Erro acessando um arquivo em #include (provavelmente o arquivo não existe)
108	Nome inapropriado para #define
109	Comando desconhecido de preprocessador (válidos #include, #define, #property, #import)
110	Ativo desconhecido para compilador
111	Função não implementada (descrição está presente, mas sem corpo)
112	Dupla aspas (") omitida
113	Chave angular de abertura (<) ou aspas duplas (") omitida
114	Aspas simples (') omitida
115	Chave angular de fechamento ">" omitida
116	Tipo não especificado na declaração
117	Sem operador return ou return não encontrado em todas as ramificações da implementação
118	Colchete de abertura de parâmetros de chamada era esperada
119	Erro escrevendo EX5
120	Acesso inválido a um array
121	A função não é do tipo void e o operador return deve retornar um valor
122	Declaração incorreta do destrutor
123	Dois pontos ":" está faltando
124	Variável já esta declarada
125	Variável com tal identificador já declarado
126	Nome de variável é longo demais (> 250 caracteres)

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
127	Estrutura com tal identificador já definida
128	Estrutura não está definida
129	Membro de estrutura com o mesmo nome já definido
130	Sem tal membro de estrutura
131	Emparelhamento de chaves violado
132	Parênteses de abertura "(" esperada
133	Chaves desbalanceadas (sem "}")
134	Dificuldade para compilar (ramificações demais, níveis de empilhamento interno são sobre-preenchidos
135	Erro de abertura de arquivo para leitura
136	Sem memória suficiente para baixar o arquivo fonte em memória
137	Variável é esperada
138	Referência não pode ser inicializada
140	Atribuição esperada (aparece na declaração)
141	Chave de abertura "{" esperada
142	Parâmetro pode ser um array dinâmico somente
143	Uso de tipo "void" não é aceitável
144	Sem par correspondente para ")" ou "]", isto é, "(" ou "[" está ausente
145	Sem par correspondente para "(" ou "[", isto é, ")" ou "]" está ausente
146	Tamanho de array incorreto
147	Excesso de parâmetros (> 64)
149	Este símbolo (token) não é esperado aqui
150	Uso inválido de operação (operandos inválidos)
151	Expressão de tipo void não é permitido
152	Operador é esperado
153	Uso incorreto do break
154	Ponto e vírgula ";" esperado
155	Vírgula "," esperada
156	Deve ser um tipo classe, não estrutura
157	Expressão é esperada

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
158	"não caractere HEX" encontrado em HEX ou número longo demais (número de dígitos > 511)
159	Constante-string tem mais que 65534 caracteres
160	Definição de função não é aceitável aqui
161	Término de programa inesperado
162	Declaração adiantada é proibida para estruturas
163	Função com este nome já está definida e tem um outro tipo de retorno
164	Função com este nome já está definida e tem um conjunto diferente de parâmetros
165	Função com este nome já está definida e implementada
166	Sobrecarga de função para esta chamada não foi encontrada
167	Função com um valor de retorno de tipo void não pode retornar um valor
168	Função não está definida
170	Valor é esperado
171	Em expressão *case* somente constantes inteiras são válidas
172	O valor de *case* neste *switch* já é usado
173	Inteiro é esperado
174	Em expressão #import, nome de arquivo é esperado
175	Expressões não são permitidas em nível global
176	Parênteses ")" omitido antes de ";"
177	Uma variável à esquerda do sinal de igualdade é esperada
178	O resultado de expressão não é usado
179	Declaração de variáveis não é permitida em *case*
180	Conversão implícita de uma string para um número
181	Conversão implícita de um número para uma string
182	Chamada ambígua de uma função sobrecarregada (várias sobrecargas se enquadram)
183	*else* ilegal sem apropriado *if*
184	Inválido *case* ou *default* sem um *switch*
185	Uso inapropriado de reticência
186	A seqüência de inicialização tem mais elementos que a variável inicializada
187	Uma constante para*case*esperada

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
188	Uma expressão constante requerida
189	Uma variável constante não pode ser alterada
190	Colchete de fechamento ou uma vírgula é esperado (declarando membro de array)
191	Identificador de enumeração já definido
192	Enumeração não pode ter modificadores de acesso (const, extern, static)
193	Membro de enumeração já declarado com um valor diferente
194	Existe uma variável definida com o mesmo nome
195	Existe uma estrutura definida com o mesmo nome
196	Nome de membro de enumeração esperado
197	Expressão de inteiro esperado
198	Divisão por zero em expressão constante
199	Número errado de parâmetros na função
200	Parâmetro por referência deve ser uma variável
201	Variável de mesmo tipo para passar por referência esperada
202	Uma variável constante não pode ser passada por uma referência não constante
203	Requer uma constante inteira positiva
204	Falha em acessar membro protegido de classe
205	Import já definido de uma outra forma
208	Arquivo executável não criado
209	Ponto de entrada 'OnCalculate' não encontrado para o indicador
210	O operador continue pode ser usado somente dentro de um loop
211	Erro acessando membro privado (fechado) de classe
213	Método de estrutura ou classe não está declarado
214	Erro acessando método privado (fechado) de classe
216	Cópia de estruturas com objetos não é permitida
218	Índice fora do intervalo do array
219	Inicialização de array em declaração de estrutura ou classe não permitida
220	Construtor de classe não pode ter parâmetros
221	Destrutor de classe não pode ter parâmetros
222	Método de classe ou estrutura com o mesmo nome e parâmetros já foi declarado

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
223	Operando esperado
224	Método de classe ou estrutura com o mesmo nome existe, mas com parâmetros diferentes (declaração!=implementação)
225	Função importada não está descrita
226	ZeroMemory() is not allowed for objects with protected members or inheritance
227	Chamada ambígua da função sobrecarregada (correspondência exata de parâmetros para várias sobrecargas)
228	Nome de variável esperada
229	Um referência não pode ser declarada neste local
230	Já usado como nome da enumeração
232	Classe ou estrutura esperada
235	Não pode chamar operador 'delete' para excluir o array
236	Operador 'while' esperado
237	Operador 'delete' deve ter um ponteiro
238	Já existe 'default' para este 'switch'
239	Erro de sintaxe
240	Seqüência-escape pode ocorrer somente em strings (começam com '\')
241	Array requerido - colchete '[' não se aplica a um array, ou não arrays são passados como parâmetros de array
242	Não pode ser inicializado através da seqüência de inicialização
243	Import não está definido
244	Erro de otimizador na árvore sintática
245	Estruturas declaradas em excesso (tente simplificar o programa)
246	Conversão do parâmetro não é permitida
247	Uso incorreto do operador 'delete'
248	Não é permitido declarar um ponteiro para uma referência
249	Não é permitido declarar uma referência para uma referência
250	Não é permitido declarar um ponteiro para um ponteiro
251	Declaração de estrutura na lista de parâmetros não é permitida
252	Operação inválida de conversão de tipo
253	Um ponteiro pode ser declarado somente para uma classe ou estrutura
256	Identificador não declarado

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
257	Erro do otimizador de código executável
258	Erro de geração de código executável
260	Expressão inválida para o operador 'switch'
261	Pool de constantes de string sobre-preenchido, simplifique o programa
262	Não pode converter para enumeração
263	Não use 'virtual' para dados (membros de uma classe ou estrutura)
264	Não pode chamar método protegido de uma classe
265	Funções virtuais sobrepostas retornam um tipo diferente
266	Classe não pode ser herdada de uma estrutura
267	Estrutura não pode ser herdada de uma classe
268	Construtor não pode ser virtual (especificador *virtual* não é permitido)
269	Estrutura não pode ter métodos virtuais
270	Função deve ter um corpo
271	Sobrecarga de funções de sistema (funções de terminal) é proibida
272	Especificador *const* não é válido para funções que não são membros de uma classe ou estrutura
274	Não permitido alterar membros de classe em método constante
276	Seqüência de inicialização inapropriada
277	Valor padrão faltante para o parâmetro (declaração específica de parâmetros default)
278	Sobreposição do parâmetro padrão (valores diferentes na declaração e implementação)
279	Não permitido chamar método não constante para um objeto constante
280	Um objeto é necessário para acessar membros (um ponteiro para uma não classe/estrutura está definido)
281	O nome de uma estrutura já declarada não pode ser usado em declaração
284	Conversão não autorizada (em herança fechada)
285	Estruturas e arrays não podem ser usados como variáveis de input
286	Especificador *const* não é válido para construtor/destrutor
287	Expressão de string incorreta para um datetime
288	Propriedade desconhecida (#property)
289	Valor incorreto de uma propriedade

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
290	Índice inválido para uma propriedade em #property
291	Chamar parâmetro omitido - <func (x,)>
293	Objeto deve ser passado por referência
294	Array deve ser passado por referência
295	Função foi declarada como exportável
296	Função não foi declarada como exportável
297	É proibido exportar funções importadas
298	Função importada não pode ter este parâmetro (proibido passar um ponteiro, classe ou estrutura contendo um array dinâmico, ponteiro, classe, etc.)
299	Deve ser uma classe
300	#import não foi fechado
302	Tipo incompatível
303	Variável externa (extern)já está inicializada
304	Nenhuma funçãoexportadaouponto de entradaencontrado
305	Chamada explícita aoconstrutornão é permitida
306	Método foi declarado comoconstante
307	Método não foi declarado comoconstante
308	Tamanho incorreto do arquivo de recurso
309	Nome de recurso incorreto
310	Erro de abertura de arquivo de recurso
311	Erro de leitura de arquivo de recurso
312	Tipo de recurso desconhecido
313	Caminho incorreto para o arquivo de recurso
314	O nome dorecursoespecificado já é usado
315	Argumento esperado para a macro tipo-função
316	Símbolo inesperado na definição de macro
317	Erro em parâmetros formais da macro
318	Número inválido de parâmetros para uma macro
319	Parâmetro em excesso para uma macro
320	Complexo demais, simplifique a macro
321	Parametro para EnumToString() somente pode ser uma enumeração

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
322	O nome do recurso é longo demais
323	Formato de imagem não suportado (somente BMP com cor de 24 ou 32 bits de profundidade é suportado)
324	Um array não pode ser declarado em operador
325	A função pode ser declarada somente no escopo global
326	A declaração não é permitida para o escopo corrente
327	Inicialização de variáveis estáticas com os valores de variáveis locais não é permitida
328	Declaração ilegal de um array de objetos que não tem um construtor padrão
329	Lista de inicialização permitida somente para construtores
330	Sem definição de função após lista de inicialização
331	Lista de inicialização está vazia
332	Inicialização de array em um construtor não é permitida
333	Inicializar membros de uma classe pai na lista de inicialização não é permitido
334	Expressão do tipo inteiro esperado
335	Memória requerida para o array excede o valor máximo
336	Memória requerida para a estrutura excede o valor máximo
337	Memória requerida para as variáveis declaradas no nível global excede o valor máximo
338	Memória requerida para variáveis locais excede o valor máximo
339	Construtornão definido
340	Nome inválido do arquivo de ícone
341	Não pode abrir o arquivo de íconeno caminho especificado
342	O arquivo de ícone está incorreto e não é do formato ICO format
343	Reinicialização de um membro em um construtor de uma classe/estrutura usando a lista de inicialização
344	Inicialização de membros estáticos na lista de inicialização do construtor não é permitida
345	Inicialização de um membro não-estático de uma classe/estrutura em nível global não é permitida
346	O nome do método da classe/estrutura corresponde ao nome de um membro declarado anteriormente

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
347	O nome do membro da classe/estrutura corresponde ao nome de um método declarado anteriormente
348	Função virtual não pode ser declarada como estática
349	O modificador const não é permitido para funções estáticas
350	Construtor ou destrutor não pode ser estático
351	Membros não-estáticos de uma classe ou estrutura não podem ser acessados a partir de uma função estática
352	Uma operação de sobrecarregada (+,-,[],++,-- etc.) é esperada após a palavra-chave operator
353	Nem todas as operações podem ser sobrecarregadas em MQL5
354	Definição não corresponde à declaração
355	Um número inválido de parâmetros é especificado para o operador
356	Função de manipulação de evento não encontrada
357	Método não pode ser exportado
358	Um ponteiro para o objeto constante não pode ser normalizado por um objeto não-constante
359	Templates de classe não são suportados ainda
360	Sobrecargade template de função não é suportado ainda
361	Template de função não pode ser aplicado
362	Parâmetro ambíguo em template de função (vários tipos de parâmetros podem ser aplicados)
363	Incapaz de determinar o tipo do parâmetro, através do qual o argumento do template de função deveria ser normalizado
364	Número incorreto de parâmetros no template de função
365	Template de função não pode ser virtual
366	Templates de função não podem ser exportados
367	Templates de função não podem ser importados
368	Estruturas contendo objetos não são permitidas
369	Arrays de seqüência e estruturas que contêm os objetos não são permitidos
370	Um membro estático de classe/estrutura deve ser explicitamente inicializado
371	Limitação do compilador: a string não pode conter mais de 65 535 caracteres
372	Inconsistente #ifdef/#endif
373	Objeto de classe não pode ser retornado, construtor de cópia não encontrado

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
374	Membros e métodos não estáticos não podem ser usados
375	OnTesterInit() impossível de usar sem o OnTesterDeinit()
376	Redefinição do parâmetro formal '%s'
377	Macro __FUNCSIG__ e __FUNCTION__ não pode aparecer fora de um corpo de função
378	Tipo inválido retornado. Por exemplo, este erro será produzido para funções importadas da DLL que retornam estrutura ou ponteiro.
379	Erro ao utilizar o modelo
380	Não é utilizado
381	Sintaxe inadmissível ao declarar apenas a função virtual, é somente permitido "=NULL" ou "=0"
382	As funções virtuais podem ser declaradas apenas com o especificador da função virtual ("=NULL" ou "=0")
383	É proibido criar um exemplar de classe abstrata
384	Para uma conversão dinâmica usando o operador dynamic_cast, o tipo de propósito deve ser o ponteiro para o tipo de usuário
385	Espera-se o tipo de "ponteiro para a função"
386	Não são suportados os ponteiros para métodos
387	Erro, não é possível determinar o tipo de ponteiro para a função
388	A conversão de tipo não está disponível devido a herançafechada
389	A variável com modificador const deve ser inicializada imediatamente após à declaração
393	Na interface podem ser declarados apenas os métodos com acesso público
394	Anexo inválido da interface para outra interface
395	A interface pode ser herdada apenas a partir de outra interface
396	À espera da interface
397	As interfaces apenas suportam herança pública
398	A interface não pode conter membros
399	Não se devem criar diretamente os objetos da interface, só através da herança
400	O especificador não pode ser usado na declaração forward
401	A herança desta classe não é possível, pois é declarada com um especificador final
402	É impossível substituir um método declarado com o especificador final
403	O especificador final só pode ser aplicado a funções virtuais

© 2000-2024, MetaQuotes Ltd.

Código	Descrição
404	Na verdade, o método marcado com o especificador override não substitui nenhuma função de classe base
405	O especificador não é permitido na definição da função, mas, sim, apenas na declaração
406	Não é possível converter esse tipo no especificado
407	Este tipo não pode ser usado para variável de recurso
408	Erro no arquivo do projeto
409	Não pode ser usado como membro union
410	Escolha ambígua para esse nome, é necessário definir claramente o contexto de uso
411	Não se pode usar essa estrutura de uma DLL
412	Não se pode chamar uma função marcada com o especificador delete
413	MQL4 não é suportado. Para compilar este programa, use o MetaEditor, que está localizado na pasta de instalação do seu terminal MetaTrader 4

© 2000-2024, MetaQuotes Ltd.

Erros em Tempo de Execução

GetLastError() é a função que retorna o código do último erro que está armazenado na variável predefinida _LastError. Este valor pode ser redefinido para zero através da função ResetLastError().

Constante	Código	Descrição
ERR_SUCCESS	0	A operação concluída com sucesso
ERR_INTERNAL_ERROR	4001	Erro interno inesperado
ERR_WRONG_INTERNAL_PARAMETER	4002	Parâmetro errado na chamada interna da função do terminal cliente
ERR_INVALID_PARAMETER	4003	Parâmetro errado ao chamar a função de sistema
ERR_NOT_ENOUGH_MEMORY	4004	Sem memória suficiente para executar a função de sistema
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	A estrutura contém objetos de strings e/ou arrays dinâmicos e/ou estrutura de tais objetos e/ou classes
ERR_INVALID_ARRAY	4006	Array de um tipo errado, tamanho errado, ou um objeto defeituoso de um array dinâmico
ERR_ARRAY_RESIZE_ERROR	4007	Sem memória suficiente para a realocação de um array, ou uma tentativa de alterar o tamanho de um array estático
ERR_STRING_RESIZE_ERROR	4008	Sem memória suficiente para a realocação de string
ERR_NOTINITIALIZED_STRING	4009	String não inicializada
ERR_INVALID_DATETIME	4010	Data e/ou hora inválida
ERR_ARRAY_BAD_SIZE	4011	Total amount of elements in the array cannot exceed 2147483647
ERR_INVALID_POINTER	4012	Ponteiro errado
ERR_INVALID_POINTER_TYPE	4013	Tipo errado de ponteiro
ERR_FUNCTION_NOT_ALLOWED	4014	Função de sistema não é permitida para chamar
ERR_RESOURCE_NAME_DUPLICATED	4015	Os nomes do recurso dinâmico e do estático equivalem
ERR_RESOURCE_NOT_FOUND	4016	Recurso com este nome não foi encontrado em EX5

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Tipo de recurso não suportado ou seu tamanho excede 16 Mb
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	O nome do recurso excede 63 caracteres
ERR_MATH_OVERFLOW	4019	Durante cálculo de funções matemáticas ocorreu um estouro
ERR_SLEEP_ERROR	4020	Excede a data de término do teste após chamar Sleep()
ERR_STOPPED	4022	Os testes foram interrompidos forçosamente de fora. Por exemplo, ao interromper a otimização, ao fechar a janela de teste visual ou ao parar o agente de teste
ERR_INVALID_TYPE	4023	Tipo inválido
ERR_INVALID_HANDLE	4024	Manipulador inválido
ERR_TOO_MANY_OBJECTS	4025	Pool de objetos cheio
Gráficos		
ERR_CHART_WRONG_ID	4101	Identificador de gráfico (chart ID) errado
ERR_CHART_NO_REPLY	4102	Gráfico não responde
ERR_CHART_NOT_FOUND	4103	Gráfico não encontrado
ERR_CHART_NO_EXPERT	4104	Nenhum Expert Advisor no gráfico que pudesse manipular o evento
ERR_CHART_CANNOT_OPEN	4105	Erro de abertura de gráfico
ERR_CHART_CANNOT_CHANGE	4106	Falha ao alterar ativo e período de um gráfico
ERR_CHART_WRONG_PARAMETER	4107	Valor de erro do parâmetro para a função que trabalha com gráficos
ERR_CHART_CANNOT_CREATE_TIMER	4108	Falha ao criar timer
ERR_CHART_WRONG_PROPERTY	4109	Identificador de propriedade (property ID) do gráfico errado
ERR_CHART_SCREENSHOT_FAILED	4110	Erro criando telas (screenshots)
ERR_CHART_NAVIGATE_FAILED	4111	Erro navegando através de gráfico
ERR_CHART_TEMPLATE_FAILED	4112	Erro aplicando template

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_CHART_WINDOW_NOT_FOUND	4113	Sub-janela contendo o indicador não foi encontrada
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Erro adicionando um indicador no gráfico
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Erro excluindo um indicador do gráfico
ERR_CHART_INDICATOR_NOT_FOUND	4116	Indicador não encontrado no gráfico especificado
Objetos Gráficos		
ERR_OBJECT_ERROR	4201	Erro trabalhando com um objeto gráfico
ERR_OBJECT_NOT_FOUND	4202	Objeto gráfico não foi encontrado
ERR_OBJECT_WRONG_PROPERTY	4203	Identificador (ID) de uma propriedade de objeto gráfico errado
ERR_OBJECT_GETDATE_FAILED	4204	Não foi possível obter data correspondente ao valor
ERR_OBJECT_GETVALUE_FAILED	4205	Não foi possível obter valor correspondente à data
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Ativo desconhecido
ERR_MARKET_NOT_SELECTED	4302	Ativo não está selecionado na janela Observação de Mercado
ERR_MARKET_WRONG_PROPERTY	4303	Identificador de uma propriedade de ativo errado
ERR_MARKET_LASTTIME_UNKNOWN	4304	Hora do último tick não é conhecida (sem ticks)
ERR_MARKET_SELECT_ERROR	4305	Erro adicionando ou excluindo um ativo na janela Observação de Marcado
Acesso de Histórico		
ERR_HISTORY_NOT_FOUND	4401	Histórico solicitado não encontrado
ERR_HISTORY_WRONG_PROPERTY	4402	Identificador (ID) da propriedade de histórico errado
ERR_HISTORY_TIMEOUT	4403	Esgotado o tempo de solicitação do histórico

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_HISTORY_BARS_LIMIT	4404	Número de barras restrito pelas configurações do terminal
ERR_HISTORY_LOAD_ERRORS	4405	Erros múltiplos ao carregar o histórico
ERR_HISTORY_SMALL_BUFFER	4407	A matriz de recebimento é muito pequena para armazenar todos os dados solicitados
Variáveis Globais		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	Variável global do terminal cliente não foi encontrada
ERR_GLOBALVARIABLE_EXISTS	4502	Variável global do terminal cliente com o mesmo nome já existe
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Não houve modificações de variáveis globais
ERR_GLOBALVARIABLE_CANNOTREAD	4504	Foi impossível abrir e ler o arquivo com os valores das variáveis globais
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	Foi impossível escrever o arquivo com os valores das variáveis globais
ERR_MAIL_SEND_FAILED	4510	Envio de email falhou
ERR_PLAY_SOUND_FAILED	4511	Reprodução de som falhou
ERR_MQL5_WRONG_PROPERTY	4512	Identificador de propriedade do programa errado
ERR_TERMINAL_WRONG_PROPERTY	4513	Identificador de propriedade do terminal errado
ERR_FTP_SEND_FAILED	4514	Envio de arquivo via ftp falhou
ERR_NOTIFICATION_SEND_FAILED	4515	Falha ao enviar uma notificação
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Parâmetro inválido para enviar uma notificação - uma string vazia ou NULL foi passada para a função SendNotification()
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Configurações de notificações erradas no terminal (ID não está especificado ou permissão não está definida)
ERR_NOTIFICATION_TOO_FREQUENT	4518	Freqüência de envio de notificações em excesso

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_FTP_NOSERVER	4519	O servidor FTP não está especificado nos atributos de configuração
ERR_FTP_NOLOGIN	4520	O login FTP não está especificado nos atributos de configuração
ERR_FTP_FILE_ERROR	4521	O ficheiro não existe
ERR_FTP_CONNECT_FAILED	4522	A ligação ao servidor FTP falhou
ERR_FTP_CHANGEDIR	4523	Não foi encontrado o diretório no servidor FTP para o upload do ficheiro
Buffers de Indicador Customizado		
ERR_BUFFERS_NO_MEMORY	4601	Sem memória suficiente para a distribuição de buffers de indicador
ERR_BUFFERS_WRONG_INDEX	4602	Índice de buffer de indicador errado
Propriedades de Indicador Customizado		
ERR_CUSTOM_WRONG_PROPERTY	4603	Identificador (ID) de propriedade do indicador customizado errado
Conta		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Identificador (ID) de propriedade da conta errado
ERR_TRADE_WRONG_PROPERTY	4751	Identificador (ID) de propriedade da negociação (trade) errado
ERR_TRADE_DISABLED	4752	Negociação via Expert Advisors proibida
ERR_TRADE_POSITION_NOT_FOUND	4753	Posição não encontrada
ERR_TRADE_ORDER_NOT_FOUND	4754	Ordem não encontrada
ERR_TRADE_DEAL_NOT_FOUND	4755	Operação (deal) não encontrada
ERR_TRADE_SEND_FAILED	4756	Envio de solicitação de negociação falhou
ERR_TRADE_CALC_FAILED	4758	Impossível calcular o valor do índice de negociação
Indicadores		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Ativo desconhecido
ERR_INDICATOR_CANNOT_CREATE	4802	Indicador não pode ser criado

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_INDICATOR_NO_MEMORY	4803	Sem memória suficiente para adicionar o indicador
ERR_INDICATOR_CANNOT_APPLY	4804	O indicador não pode ser aplicado a um outro indicador
ERR_INDICATOR_CANNOT_ADD	4805	Erro ao aplicar um indicador ao gráfico
ERR_INDICATOR_DATA_NOT_FOUND	4806	Dado solicitado não encontrado
ERR_INDICATOR_WRONG_HANDLE	4807	Manuseio de indicador errado
ERR_INDICATOR_WRONG_PARAMETERS	4808	Numero errado de parâmetros ao criar um indicador
ERR_INDICATOR_PARAMETERS_MISSING	4809	Sem parâmetros ao criar um indicador
ERR_INDICATOR_CUSTOM_NAME	4810	O primeiro parâmetro no array deve ser o nome do indicador customizado
ERR_INDICATOR_PARAMETER_TYPE	4811	Tipo de parâmetro inválido no array ao criar um indicador
ERR_INDICATOR_WRONG_INDEX	4812	Índice errado de buffer do indicador solicitado
Profundidade de Mercado		
ERR_BOOKS_CANNOT_ADD	4901	Profundidade de Mercado não pode ser adicionado
ERR_BOOKS_CANNOT_DELETE	4902	Profundidade de Mercado não pode ser removido
ERR_BOOKS_CANNOT_GET	4903	Os dados da Profundidade de Mercado não podem ser obtidos
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Erro em subscrever para receber novos dados da Profundidade de Mercado
Operações de Arquivo		
ERR_TOO_MANY_FILES	5001	Mais que 64 arquivos não podem ser abertos ao mesmo tempo
ERR_WRONG_FILENAME	5002	Nome de arquivo inválido
ERR_TOO_LONG_FILENAME	5003	Nome de arquivo longo demais
ERR_CANNOT_OPEN_FILE	5004	Erro de abertura de arquivo
ERR_FILE_CACHEBUFFER_ERROR	5005	Sem memória suficiente de cache para leitura

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_CANNOT_DELETE_FILE	5006	Erro excluindo arquivo
ERR_INVALID_FILEHANDLE	5007	Um arquivo com este handle foi fechado, ou simplesmente não estava aberto
ERR_WRONG_FILEHANDLE	5008	Handle de arquivo errado
ERR_FILE_NOTTOWRITE	5009	O arquivo deve estar abertura para escrita
ERR_FILE_NOTTOREAD	5010	O arquivo deve estar aberto para leitura
ERR_FILE_NOTBIN	5011	O arquivo deve estar aberto como um arquivo binário
ERR_FILE_NOTTXT	5012	O arquivo deve estar aberto como um texto
ERR_FILE_NOTTXTORCSV	5013	O arquivo deve estar aberto como um texto ou CSV
ERR_FILE_NOTCSV	5014	O arquivo deve estar aberto como CSV
ERR_FILE_READERROR	5015	Erro de leitura de arquivo
ERR_FILE_BINSTRINGSIZE	5016	Tamanho da string deve estar especificado, porque o arquivo está aberto como binário
ERR_INCOMPATIBLE_FILE	5017	Um arquivo de texto deve ser usado para arrays de strings, para outros arrays - binários
ERR_FILE_IS_DIRECTORY	5018	Isto não é um arquivo, isto é um diretório
ERR_FILE_NOT_EXIST	5019	Arquivo inexistente
ERR_FILE_CANNOT_REWRITE	5020	Arquivo não pode ser reescrito
ERR_WRONG_DIRECTORYNAME	5021	Nome de diretório errado
ERR_DIRECTORY_NOT_EXIST	5022	Diretório inexistente
ERR_FILE_ISNOT_DIRECTORY	5023	Isto é um arquivo, não um diretório
ERR_CANNOT_DELETE_DIRECTORY	5024	O diretório não pode ser removido
ERR_CANNOT_CLEAN_DIRECTORY	5025	Falha ao limpar o diretório (provavelmente um ou mais arquivos estão bloqueados e a operação de remoção falhou)

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_FILE_WRITEERROR	5026	Falha ao escrever um recurso para um arquivo
ERR_FILE_ENDOFFILE	5027	Falha ao ler o próximo bloco de dados a partir do arquivo CSV (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool), porque atingido o final do arquivo
Conversão de String		
ERR_NO_STRING_DATE	5030	Sem data na string
ERR_WRONG_STRING_DATE	5031	Data errada na string
ERR_WRONG_STRING_TIME	5032	Hora errada na string
ERR_STRING_TIME_ERROR	5033	Erro convertendo string em data
ERR_STRING_OUT_OF_MEMORY	5034	Sem memória suficiente para a string
ERR_STRING_SMALL_LEN	5035	O comprimento da string é menor que o esperado
ERR_STRING_TOO_BIGNUMBER	5036	Número grande demais, maior que ULONG_MAX
ERR_WRONG_FORMATSTRING	5037	string de formato inválido
ERR_TOO_MANY_FORMATTERS	5038	Quantidade de especificadores de formato maior que de parâmetros
ERR_TOO_MANY_PARAMETERS	5039	Quantidade de parâmetros maior que de especificadores de formato
ERR_WRONG_STRING_PARAMETER	5040	Parâmetro de tipo string defeituoso
ERR_STRINGPOS_OUTOFRANGE	5041	Posição fora da string
ERR_STRING_ZEROADDED	5042	0 adicionado ao final da string, uma operação inútil
ERR_STRING_UNKNOWNTYPE	5043	Tipo de dado desconhecido ao converter para uma string
ERR_WRONG_STRING_OBJECT	5044	Objeto de string defeituoso
Operações com Arrays		
ERR_INCOMPATIBLE_ARRAYS	5050	Copiando arrays incompatíveis. Array de string pode ser copiado somente para um array de string,

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
		e um array numérico - somente em um array numérico
ERR_SMALL_ASSERIES_ARRAY	5051	O array de recepção está declarado como AS_SERIES, e é de tamanho insuficiente.
ERR_SMALL_ARRAY	5052	Array pequeno demais, a posição inicial está fora do array
ERR_ZEROSIZE_ARRAY	5053	Um array de comprimento zero
ERR_NUMBER_ARRAYS_ONLY	5054	Deve ser um array numérico
ERR_ONEDIM_ARRAYS_ONLY	5055	Deve ser um array unidimensional
ERR_SERIES_ARRAY	5056	Série de tempo (timeseries) não pode ser usada
ERR_DOUBLE_ARRAY_ONLY	5057	Deve ser um array de tipo double
ERR_FLOAT_ARRAY_ONLY	5058	Deve ser um array de tipo float
ERR_LONG_ARRAY_ONLY	5059	Deve ser um array de tipo long
ERR_INT_ARRAY_ONLY	5060	Deve ser um array de tipo int
ERR_SHORT_ARRAY_ONLY	5061	Deve ser um array de tipo short
ERR_CHAR_ARRAY_ONLY	5062	Deve ser um array de tipo char
ERR_STRING_ARRAY_ONLY	5063	Deve ser uma matriz do tipo cadeia de caracteres
Operações com OpenCL		
ERR_OPENCL_NOT_SUPPORTED	5100	Funções OpenCL não são suportados neste computador
ERR_OPENCL_INTERNAL	5101	Erro interno ocorreu ao executar OpenCL
ERR_OPENCL_INVALID_HANDLE	5102	Handle de OpenCL inválido
ERR_OPENCL_CONTEXT_CREATE	5103	Erro criando o contexto de OpenCL
ERR_OPENCL_QUEUE_CREATE	5104	Falha ao criar um fila de execução em OpenCL
ERR_OPENCL_PROGRAM_CREATE	5105	Erro ocorreu ao compilar um programa OpenCL
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Nome de kernel longo demais(kernel de OpenCL)
ERR_OPENCL_KERNEL_CREATE	5107	Erro criando um kernel de OpenCL

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	Erro ocorreu ao configurar parâmetrospara o kernel de OpenCL
ERR_OPENCL_EXECUTE	5109	Erro em tempo de execução de programa de OpenCL
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	Tamanho inválido do buffer de OpenCL
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	Deslocamento (offset) inválido do buffer de OpenCL
ERR_OPENCL_BUFFER_CREATE	5112	Falha ao criar um buffer de OpenCL
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Excedido o número máximo de objetos OpenCL
ERR_OPENCL_SELECTDEVICE	5114	Erro ao selecionar o dispositivo OpenCL
Trabalhar com bancos de dados		
ERR_DATABASE_INTERNAL	5120	Erro interno no banco de dados
ERR_DATABASE_INVALID_HANDLE	5121	Identificador de banco de dados inválido
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Número máximo de objetos de banco de dados excedido
ERR_DATABASE_CONNECT	5123	Erro ao conectar ao banco de dados
ERR_DATABASE_EXECUTE	5124	Erro de execução de consulta
ERR_DATABASE_PREPARE	5125	Erro ao criar consulta
ERR_DATABASE_NO_MORE_DATA	5126	Não há mais dados de leitura
ERR_DATABASE_STEP	5127	Erro ao passar para o próximo registro da consulta
ERR_DATABASE_NOT_READY	5128	Os dados para leitura dos resultados da consulta ainda não estão prontos
ERR_DATABASE_BIND_PARAMETERS	5129	Erro de substituição automática de parâmetro na consulta SQL
Trabalhando com WebRequest()		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	URL inválido
ERR_WEBREQUEST_CONNECT_FAILED	5201	Falha ao conectar com o URL especificado

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_WEBREQUEST_TIMEOUT	5202	Tempo limite excedido
ERR_WEBREQUEST_REQUEST_FAILED	5203	Falha na solicitação do HTTP
Trabalhando com uma rede (soquetes)		
ERR_NETSOCKET_INVALIDHANDLE	5270	Para a função foi passado um identificador de soquete inválido.
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Muitos soquetes abertos (máximo 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Erro ao conectar-se ao host remoto
ERR_NETSOCKET_IO_ERROR	5273	Erro ao enviar/receber dados do soquete
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Erro ao estabelecer uma conexão segura (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	Não há dados sobre o certificado que protege a conexão.
Símbolos personalizados		
ERR_NOT_CUSTOM_SYMBOL	5300	Deve ser especificado o símbolo personalizado
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Nome de símbolo personalizado incorreto. No nome do símbolo é possível utilizar apenas caracteres latinos sem pontuação, espaços e caracteres especiais (são permitidos ".", "_", "&" e "#"). Não é recomendado usar os caracteres <, >, :, ", /,\, |, ?, *.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Nome demasiado longo para o símbolo personalizado. O nome não pode exceder 32 caracteres tendo em conta o 0 final
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Caminho demasiado longo para o símbolo personalizado. A caminho não deve exceder 128 caracteres, tendo em conta "Custom\\", o nome do símbolo, separadores de grupos e o 0 final
ERR_CUSTOM_SYMBOL_EXIST	5304	Já existe um caráter personalizado com esse nome
ERR_CUSTOM_SYMBOL_ERROR	5305	Erro ao criar, excluir ou modificar o símbolo personalizado

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_CUSTOM_SYMBOL_SELECTED	5306	Tentativa de excluir um símbolo personalizado selecionado na Observação do mercado (Market Watch)
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Propriedade de símbolo personalizado incorreto
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Parâmetro errado ao definir propriedade do símbolo personalizado
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Parâmetro de cadeia muito longo ao definir propriedade do símbolo personalizado
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	Array de ticks não sequenciada por tempo
Calendário econômico		
ERR_CALENDAR_MORE_DATA	5400	Tamanho de array insuficiente para obter descrições de todos os valores
ERR_CALENDAR_TIMEOUT	5401	Limite de tempo da solicitação excedido
ERR_CALENDAR_NO_DATA	5402	País não encontrado
Working with databases		
ERR_DATABASE_ERROR	5601	Generic error
ERR_DATABASE_LOGIC	5602	SQLite internal logic error
ERR_DATABASE_PERM	5603	Access denied
ERR_DATABASE_ABORT	5604	Callback routine requested abort
ERR_DATABASE_BUSY	5605	Database file locked
ERR_DATABASE_LOCKED	5606	Database table locked
ERR_DATABASE_NOMEM	5607	Insufficient memory for completing operation
ERR_DATABASE_READONLY	5608	Attempt to write to readonly database
ERR_DATABASE_INTERRUPT	5609	Operation terminated by sqlite3_interrupt()
ERR_DATABASE_IOERR	5610	Disk I/O error
ERR_DATABASE_CORRUPT	5611	Database disk image corrupted

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_DATABASE_NOTFOUND	5612	Unknown operation code in sqlite3_file_control()
ERR_DATABASE_FULL	5613	Insertion failed because database is full
ERR_DATABASE_CANTOPEN	5614	Unable to open the database file
ERR_DATABASE_PROTOCOL	5615	Database lock protocol error
ERR_DATABASE_EMPTY	5616	Internal use only
ERR_DATABASE_SCHEMA	5617	Database schema changed
ERR_DATABASE_TOOBIG	5618	String or BLOB exceeds size limit
ERR_DATABASE_CONSTRAINT	5619	Abort due to constraint violation
ERR_DATABASE_MISMATCH	5620	Data type mismatch
ERR_DATABASE_MISUSE	5621	Library used incorrectly
ERR_DATABASE_NOLFS	5622	Uses OS features not supported on host
ERR_DATABASE_AUTH	5623	Authorization denied
ERR_DATABASE_FORMAT	5624	Not used
ERR_DATABASE_RANGE	5625	Bind parameter error, incorrect index
ERR_DATABASE_NOTADB	5626	File opened that is not database file
Métodos matriciais e vetoriais		
ERR_MATRIX_INTERNAL	5700	Erro interno do subsistema de execução de matrizes/vetores
ERR_MATRIX_NOT_INITIALIZED	5701	Matriz/vetor não inicializado
ERR_MATRIX_INCONSISTENT	5702	Tamanho de matrizes/vetores em operação inconsistente
ERR_MATRIX_INVALID_SIZE	5703	Tamanho da matriz/vetor incorreto
ERR_MATRIX_INVALID_TYPE	5704	Tipo de matriz/vetor inválido
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	Função não disponível para esta matriz/vetor
ERR_MATRIX_CONTAINS_NAN	5706	Matriz/vetor contém não-números (Nan/Inf)
Modelos ONNX		

© 2000-2024, MetaQuotes Ltd.

Constante	Código	Descrição
ERR_ONNX_INTERNAL	5800	Erro interno do padrão ONNX
ERR_ONNX_NOT_INITIALIZED	5801	Erro de inicialização ONNX Runtime API
ERR_ONNX_NOT_SUPPORTED	5802	Propriedade ou valor não compatível com a linguagem MQL5
ERR_ONNX_RUN_FAILED	5803	Erro ao iniciar a ONNX runtime API
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	Número inválido de parâmetros passados para OnnxRun
ERR_ONNX_INVALID_PARAMETER	5805	Valor de parâmetro incorreto
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	Tipo de parâmetro incorreto
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	Tamanho de parâmetro incorreto
ERR_ONNX_WRONG_DIMENSION	5808	Dimensão do tensor não definida ou especificada incorretamente
ERR_USER_ERROR_FIRST	65536	Erros definidos pelo usuário começam com este código

Veja Também

Códigos de Retorno do Servidor de Negociação

© 2000-2024, MetaQuotes Ltd.

Constantes de Entrada e Saída

Constantes:

- Flags de abertura de arquivo
- Propriedades de arquivo
- Posicionamento dentro de um arquivo
- Uso de code page
- MessageBox

© 2000-2024, MetaQuotes Ltd.

Flags de Abertura de Arquivo

Valores de flags de abertura de arquivo especificam o modo de acesso do arquivo. Flags são definidos como se segue:

Identificador	Valor	Descrição
FILE_READ	1	Arquivo é aberto para leitura. Flag é usado em FileOpen(). Ao abrir um arquivo, a especificação de FILE_WRITE e/ou FILE_READ é requerida.
FILE_WRITE	2	Arquivo é aberto para escrita. Flag é usado em FileOpen(). Ao abrir um arquivo, a especificação de FILE_WRITE e/ou FILE_READ é requerida.
FILE_BIN	4	Modo de leitura/escrita binária (sem conversão string para string). Flag é usado em FileOpen()
FILE_CSV	8	Arquivo CSV (todos seus elementos são convertidos para strings do tipo apropriado, unicode ou ansi, e separados por separador). Flag é usado em FileOpen()
FILE_TXT	16	Arquivo de texto simples (semelhando ao arquivo csv, mas sem levar em conta os separadores). Flag é usado em FileOpen()
FILE_ANSI	32	Strings de tipo ANSI (símbolos de um byte). Flag é usado em FileOpen()
FILE_UNICODE	64	Strings de tipo UNICODE (símbolos de dois bytes). Flag é usado em FileOpen()
FILE_SHARE_READ	128	Acesso compartilhado para leitura a partir de vários programas. Flag é usado em FileOpen(), mas ele não substitui a necessidade de indicar o flag FILE_WRITE e/ou FILE_READ ao abrir um arquivo.
FILE_SHARE_WRITE	256	Acesso compartilhado para escrita a partir de vários programas. Flag é usado em FileOpen(), mas ele não substitui a necessidade de indicar o flag FILE_WRITE e/ou FILE_READ ao abrir um arquivo.
FILE_REWRITE	512	Possibilidade de reescrever o arquivo usando as funções FileCopy() e FileMove(). O arquivo deve existir ou deve estar aberto para escrita, de outra forma o arquivo não será aberto.
FILE_COMMON	4096	O caminho do arquivo na pasta comum de todos os terminais cliente \Terminal\Common\Files. Flag é usado em funções FileOpen(), FileCopy(), FileMove() e em FileIsExist().

Um ou vários flags podem ser especificado ao abrir um arquivo. Esta é uma combinação de flags. A combinação de flags é escrita usando o sinal de lógica OR (|), que é posicionado entre os flags

© 2000-2024, MetaQuotes Ltd.

enumerados. Por exemplo, para abrir um arquivo em formato CSV para leitura e escrita ao mesmo tempo, especifique a combinação FILE_READ|FILE_WRITE|FILE_CSV.

Exemplo:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Existem algumas características específicas de funcionamento ao se especificar flags de leitura e escrita:

- Se FILE_READ é especificado, uma tentativa é feita para abrir um arquivo existente. Caso o arquivo não exista, a abertura do arquivo falha, um novo arquivo não é criado.
- FILE_READ|FILE_WRITE – um novo arquivo é criado caso o arquivo com o nome especificado não exista.
- FILE_WRITE – o arquivo é criado novamente com tamanho zero.

Ao abrir um arquivo, a especificação de FILE_WRITE e/ou FILE_READ é requerida.

Flags que definem o tipo de leitura de um arquivo aberto, possuem prioridade. O flag de mais alta prioridade é FILE_CSV, daí vem FILE_BIN, e FILE_TXT é o de mais baixa prioridade. Assim, se vários flags são especificados ao mesmo tempo (FILE_TXT|FILE_CSV ou FILE_TXT|FILE_BIN ou FILE_BIN|FILE_CSV), o flag com a mais alta prioridade será usado.

Flags que definem o tipo de codificação também têm prioridade. FILE_UNICODE é de maior prioridade que FILE_ANSI. Assim, se você especificar a combinação FILE_UNICODE|FILE_ANSI, o flag FILE_UNICODE será usado.

Se nem FILE_UNICODE ou FILE_ANSI forem indicados, FILE_UNICODE fica implícito. Se nem FILE_CSV, ou FILE_BIN, ou FILE_TXT forem especificados, FILE_CSV fica implícito.

Se um arquivo é aberto para leitura como um arquivo de texto (FILE_TXT ou FILE_CSV), e no início do arquivo uma indicação de dois-bytes `0xff,0xfe` é encontrada, o flag de codificação será FILE_UNICODE, mesmo se FILE_ANSI for especificado.

Também Veja

Funções de Arquivo

© 2000-2024, MetaQuotes Ltd.

Propriedades de Arquivo

A função FileGetInteger() é usada para obter propriedades de arquivo. O identificador da propriedade requerida proveniente da enumeração ENUM_FILE_PROPERTY_INTEGER é passado para ela durante a chamada.

ENUM_FILE_PROPERTY_INTEGER

ID	Descrição
FILE_EXISTS	Verifica a existência
FILE_CREATE_DATE	Data de criação
FILE_MODIFY_DATE	Data da última modificação
FILE_ACCESS_DATE	Data do último acesso ao arquivo
FILE_SIZE	Tamanho de arquivo em bytes
FILE_POSITION	Posição de um ponteiro no arquivo
FILE_END	Obtém o sinal de término de arquivo
FILE_LINE_END	Obtém o sinal de termino de linha
FILE_IS_COMMON	O arquivo está aberto em uma pasta compartilhada por todos os terminais (veja FILE_COMMON)
FILE_IS_TEXT	O arquivo está aberto como um arquivo de texto (veja FILE_TXT)
FILE_IS_BINARY	O arquivo está aberto como um arquivo binário (veja FILE_BIN)
FILE_IS_CSV	O arquivo está aberto como CSV (veja FILE_CSV)
FILE_IS_ANSI	O arquivo está aberto como ANSI (veja FILE_ANSI)
FILE_IS_READABLE	O arquivo aberto pode ser lido (veja FILE_READ)
FILE_IS_WRITABLE	O arquivo aberto pode ser escrito (veja FILE_WRITE)

A função FileGetInteger() tem duas opções diferentes de chamada. Na primeira opção, para obter propriedades de um arquivo, seu handle é especificado, que é obtido na abertura do arquivo usando a função FileOpen(). Esta opção permite obter todas as propriedades de um arquivo.

A segunda opção da função FileGetInteger() retorna valores de propriedades de arquivo através do nome do arquivo. Usando esta opção, somente as seguintes propriedades gerais podem ser obtidas:

- FILE_EXISTS – existência de um arquivo com o nome especificado
- FILE_CREATE_DATE – data de criação do arquivo com o nome especificado
- FILE_MODIFY_DATE – data de modificação do arquivo com o nome especificado
- FILE_ACCESS_DATE - data do último acesso ao arquivo com o nome especificado
- FILE_SIZE – tamanho do arquivo com o nome especificado

Ao tentar obter propriedades outras daquelas especificadas acima, a segunda opção de chamada de FileGetInteger() retornará um erro.

© 2000-2024, MetaQuotes Ltd.

Posicionamento Dentro de um Arquivo

A maioria das funções de arquivo é associada com operações leitura/escrita de dados. Ao mesmo tempo, usando a função FileSeek() você pode especificar a posição de um ponteiro de arquivo para uma posição dentro do arquivo, a partir da qual a próxima operação de leitura ou escrita será realizada. A enumeração ENUM_FILE_POSITION contém posições válidas de ponteiro, com a qual você pode especificar um deslocamento em bytes para a próxima operação.

ENUM_FILE_POSITION

Identificador	Descrição
SEEK_SET	Início de arquivo
SEEK_CUR	Posição corrente de um ponteiro de arquivo
SEEK_END	Fim de arquivo

Também Veja

FileIsEnding, FileIsLineEnding

© 2000-2024, MetaQuotes Ltd.

Usando um Código de Página (Codepage) em Operações de Conversão de String

Ao converter variáveis string em arrays de tipo char e vice-versa, a codificação que por default correspodente ao ANSI corrente do sistema operacional Windows (CP_ACP) é usada em MQL5. Se você quiser especificar um tipo diferente de codificação, isso pode ser definido como parâmetro adicional para as funções CharArrayToString(), StringToCharArray() e FileOpen().

A tabela lista as constantes internas de alguns dos mais populares códigos de página. Códigos de página não mencionados podem ser especificados através de um código correspondente à página.

Constantes Internas de Código de Páginas (Codepages)

Constante	Valor	Descrição
CP_ACP	0	O código de página ANSI do Windows corrente.
CP_OEMCP	1	O código de página do sistema OEM corrente.
CP_MACCP	2	O código de página do sistema Macintosh corrente. **Note:** Este valor é usado principalmente em códigos de programa criados antigamente e está sem uso atualmente, já que computadores Macintosh modernos usando Unicode para codificação.
CP_THREAD_ACP	3	O código de página ANSI do Windows para o thread corrente.
CP_SYMBOL	42	Código de página de símbolo
CP_UTF7	65000	Código de página UTF-7.
CP_UTF8	65001	Código de página UTF-8.

Também Veja

Propriedades do Terminal Cliente

© 2000-2024, MetaQuotes Ltd.

Constantes da Janela de Diálogo MessageBox

Esta seção contém códigos de retorno da função MessageBox(). Se um janela de mensagem tem um botão Cancel, a função retorna IDCANCEL no caso da tecla ESC ou o botão Cancel serem pressionados. Caso não exista nenhum botão Cancel na janela de mensagem, pressionar o ESC não produz nenhum efeito.

Constante	Valor	Descrição
IDOK	1	Botão "OK" foi pressionado
IDCANCEL	2	Botão "Cancel" foi pressionado
IDABORT	3	Botão "Abort" foi pressionado
IDRETRY	4	Botão "Retry" foi pressionado
IDIGNORE	5	Botão "Ignore" foi pressionado
IDYES	6	Botão "Yes" foi pressionado
IDNO	7	Botão "No" foi pressionado
IDTRYAGAIN	10	Botão "Try Again" foi pressionado
IDCONTINUE	11	Botão "Continue" foi pressionado

Os principais flags da função MessageBox() definem o conteúdo e comportamento da janela de diálogo. Este valor pode ser uma combinação dos seguintes grupos de flag:

Constante	Valor	Descrição
MB_OK	0x00000000	Janela de mensagem contém somente um botão: OK. Default
MB_OKCANCEL	0x00000001	Janela de mensagem contém dois botões: OK e Cancel
MB_ABORTRETRYIGNORE	0x00000002	Janela de mensagem contém três botões: Abort, Retry e Ignore
MB_YESNOCANCEL	0x00000003	Janela de mensagem contém três botões: Yes, No e Cancel
MB_YESNO	0x00000004	Janela de mensagem contém dois botões: Yes e No
MB_RETRYCANCEL	0x00000005	Janela de mensagem contém dois botões: Retry e Cancel
MB_CANCELTRYCONTINUE	0x00000006	Janela de mensagem contém três botões: Cancel, Try Again, Continue

Para exibir um ícone na janela de mensagem é necessário especificar flags adicionais:

© 2000-2024, MetaQuotes Ltd.

Constante	Valor	Descrição
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	O ícone de sinal STOP
MB_ICONQUESTION	0x00000020	O ícone de sinal de interrogação
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	O ícone de sinal de exclamação/aviso
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	O sinal i cercado

Os botões default são definidos através dos seguintes flags:

Constante	Valor	Descrição
MB_DEFBUTTON1	0x00000000	O primeiro botão MB_DEFBUTTON1 - é o default, se os outros botões MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 não forem especificados
MB_DEFBUTTON2	0x00000100	O segundo botão é o default
MB_DEFBUTTON3	0x00000200	O terceiro botão é o default
MB_DEFBUTTON4	0x00000300	O quarto botão é o default

© 2000-2024, MetaQuotes Ltd.

Programas MQL5

Para um programa-mql5 funcionar, ele deve ser compilado (botão Compilar ou tecla F7). A compilação deve passar sem erros (alguns avisos são possíveis; eles devem ser analisados). Neste processo, um arquivo executável com o mesmo nome e com extensão EX5 deve ser criado no diretório correspondente, terminal_dir\MQL5\Experts, terminal_dir\MQL5\indicators ou terminal_dir\MQL5\scripts. Este arquivo pode ser executado.

Características operacionais de programas MQL5 são descritas nas seguintes seções:

- Execução de Programa – ordem de chamada de manipuladores de eventos predefinidos.
- Testando estratégias de negociação – características operacionais de programas MQL5 no Provador de Estratégias.
- Eventos do terminal cliente – descrição de eventos, que podem ser processados em programas.
- Chamada de funções importadas – ordem de descrição, parâmetros permitidos, detalhes de busca e convenção de chamada para funções importadas.
- Erros em tempo de execução – obtenção de informações sobre erros críticos e em tempo de execução.

Expert Advisors, indicadores customizados e scripts são anexados a um dos gráficos abertos através do método "Arrastar e Soltar" a partir da janela Navegador.

Para um Expert Advisor parar de funcionar, ele deve ser removido de um gráfico. Para fazer isso selecione a "Lista de Experts" no menu de contexto do gráfico, então selecione um Expert Advisor da lista e clique no botão "Remover". O funcionamento de Expert Advisors é também afetado pelo estado do botão "AutoTrading".

A fim de parar um indicador customizado, ele deve ser removido de um gráfico.

Indicadores customizados e Expert Advisors trabalham até eles serem explicitamente removidos de um gráfico; informações sobre Expert Advisors e Indicadores são salvos entre sessões do terminal cliente.

Scripts são executados uma vez e são excluídos automaticamente na conclusão da operação ou mudança de estado do gráfico corrente, ou no encerramento do terminal cliente. Após o reinício do terminal cliente, scripts não são iniciados, porque as informações sobre eles não são salvas.

Em um gráfico podem operar no máximo um Expert Advisor, um script e um número ilimitado de indicadores.

Os serviços não precisam estar vinculados ao gráfico para funcionarem e são projetados para executar funções auxiliares. Por exemplo, no serviço você pode criar um símbolo personalizado, abrir o gráfico do símbolo criado e, em um loop infinito, obter dados para ele usando funções de rede e atualizar continuamente.

© 2000-2024, MetaQuotes Ltd.

Execução de Programa

Cada script, serviço e EA é executado em seu próprio fluxo separado. Todos os indicadores calculados com base em um símbolo, mesmo se estiverem sendo executados em gráficos diferentes, funcionam no mesmo fluxo. Assim, todos os indicadores em um símbolo compartilham os recursos de um fluxo.

Todas as outras ações associadas a um símbolo, como processamento de ticks e sincronização de histórico, também são consistentemente realizadas no mesmo segmento com indicadores. Isso significa que se uma ação infinita é realizada em um indicador, todos os outros eventos associados a seu ativo nunca serão realizados.

Ao executar um Expert Advisor, certifique-se de que ele tem um ambiente de negociação real e pode acessar o histórico do requerido ativo e período, e sincronizar dados entre o terminal e o servidor. Para todos estes procedimentos, o terminal fornece um atraso de iniciação de não mais que 5 segundos, após o qual o Expert Advisor será iniciado com os dados disponíveis. Portanto, no caso de não haver conexão com o servidor, isso pode levar a um atraso na iniciação de um Expert Advisor.

A tabela abaixo contém um breve resumo sobre programas MQL5:

Programa	Execução	Observação
Serviço	No fluxo em si, há o mesmo número de serviços e de fluxos de execução para eles	O serviço em loop não pode atrapalhar outros programas
Script	Um segmento separado, o número de segmentos para scripts é igual ao número de scripts	Um script em loop não é capaz de interromper a execução de outros programas
Expert Advisor	Um segmento separado, o número de segmentos para Expert Advisor é igual ao número de Expert Advisor	Um Expert Advisor em loop não é capaz de interromper a execução de outros programas
Indicador	Um segmento para todos os indicadores de um símbolo. O número de segmentos é igual ao número de símbolos com indicadores	Um loop infinito em um indicador irá parar todos os outros indicadores sobre este ativo

Logo após um programa ser anexado a um gráfico, ele é carregado para a memória do terminal cliente, bem como variáveis globais são inicializadas. Caso alguma variável global do tipo classe tiver um construtor, este construtor será chamado durante a inicialização das variáveis globais.

Após isso, o programa fica esperando por um evento do terminal cliente. Cada programa-MQL5 deve ter ao menos um manipulador (handler) de evento, caso contrário o programa carregado não será executado. Handlers de evento têm nomes pré-definidos, parâmetros e tipos de retorno.

© 2000-2024, MetaQuotes Ltd.

Tipo	Nome da Função	Parâmetros	Aplicativo	Comentário
int	OnInit	nenhum	Expert Advisors e indicadores	Handler de evento Init. Ele permite usar o tipo de retorno void.
void	OnDeinit	const int reason	Expert Advisors e indicadores	Handler de evento Deinit.
void	OnStart	nenhum	scripts e serviços	Handler de evento Start.
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	indicadores	Handler de evento Calculate para todos os preços.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	indicadores	Handler de evento Calculate sobre um array único de dados. Um indicador não pode ter simultaneamente dois handlers de evento. Neste caso apenas um handler (manipulador) de evento vai trabalhar na array de dados.
void	OnTick	nenhum	Expert Advisors	Handler de evento NewTick. Enquanto o evento de recebimento de um novo tick está sendo processado, nenhum outro evento deste tipo é recebido.
void	OnTimer	nenhum	Expert Advisors e indicadores	Handler de evento Timer.

© 2000-2024, MetaQuotes Ltd.

Tipo	Nome da Função	Parâmetros	Aplicativo	Comentário
void	OnTrade	nenhum	Expert Advisors	Handler de evento Trade.
double	OnTester	nenhum	Expert Advisors	Handler de evento Tester.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Expert Advisors e indicadores	Handler de evento ChartEvent.
void	OnBookEvent	const string &symbol_name	Expert Advisors e indicadores	Handler de evento BookEvent.

Um terminal cliente envia novos eventos para o gráfico aberto correspondente. Os eventos podem ser gerados por gráficos (Eventos de gráfico) ou programas-MQL5 (eventos customizados). A geração de eventos de criação ou exclusão de objetos gráficos sobre um gráfico pode ser habilitado ou desabilitado através da definição das propriedades de gráfico CHART_EVENT_OBJECT_CREATE e CHART_EVENT_OBJECT_DELETE. Cada programa MQL5 e cada gráfico têm sua própria fila de eventos, onde todos os novos eventos que chegam são adicionados.

Um programa recebe somente eventos do gráfico no qual ele está rodando. Todos os eventos são processados um após o outro na ordem que em eles são recebidos. Se um fila já tiver um evento NewTick, ou este evento está correntemente sendo processado, então um novo evento NewTick não é colocado na fila do programa MQL5. Similarmente, se ChartEvent já esta enfileirado, ou este evento está sendo processado, nenhum novo evento deste tipo é enfileirado. Os eventos de timer são tratados da mesma forma – se o evento Timer está na filha ou sendo tratado, um novo evento timer não é enfileirado .

Filas de evento têm um tamanho limitado mas suficiente, de modo que a sobrecarga de uma fila em programas bem escritos é improvável. No caso de uma sobrecarga de fila, novos eventos são descartados sem enfileiramento.

Evite usar loops infinitos para manipular eventos. As únicas exceções a essa regra são scripts e serviços que manipulam um único evento Start.

Bibliotecas não tratam nenhum evento.

Funções proibidas em Indicadores e Expert Advisors

Indicatores, scripts e Expert Advisors são programas executáveis escritos em MQL5. Eles são projetados para diferentes tipos de tarefas. Portanto existem algumas restrições quanto ao uso de certas funções, dependendo do tipo de programa. As seguintes funções são proibidas em indicadores:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();

© 2000-2024, MetaQuotes Ltd.

- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

Todas as funções projetadas para indicadores são proibidas em Expert Advisors e scripts:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

A biblioteca (library) não é um programa independente e é executado no contexto do programa MQL5 que a chamou: script, indicador ou Expert Advisor. Desta forma, as restrições acima se aplicam à biblioteca (library) chamada.

Proibição do uso de funções nos serviços

Os serviços não aceitam eventos, pois não estão vinculados a gráficos. Nos serviços são proibidas as seguintes funções:

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Carga e Descarga de Indicadores

© 2000-2024, MetaQuotes Ltd.

Indicadores são carregados nos seguintes casos:

- um indicador é anexado a um gráfico;
- inicio do terminal (se o indicador estava anexado a um gráfico antes do encerramento do terminal);
- carga de um template (se o indicador anexado a um gráfico estiver especificado no template);
- alteração de um perfil (se o indicador estiver anexado a um dos gráficos do perfil alterado);
- alteração de um ativo e/ou período de um gráfico, no qual o indicador está anexado;
- alteração da conta na qual o terminal está conectado;
- após uma bem sucedida recompilação de um indicador, se o indicador estiver anexado a um gráfico;
- alteração de parâmetros de entrada do indicador.

Indicadores são descarregados nos seguintes casos:

- ao desanexar um indicador de um gráfico;
- encerramento do terminal (se o indicador estiver anexado a um gráfico);
- carga de um template, se um indicador estiver anexado a um gráfico;
- fechamento de um gráfico, no qual o indicador estava anexado;
- alteração de um perfil, se o indicador estiver anexado a um dos gráficos do perfil alterado;
- alteração de um ativo e/ou período de um gráfico, no qual o indicador está anexado;
- alteração da conta na qual o terminal está conectado;
- alteração de parâmetros de entrada do indicador.

Carga e Descarga de Expert Advisors

Expert Advisors são carregados nos seguintes casos:

- ao anexar um Expert Advisor a um gráfico;
- início de terminal (se o Expert Advisor estava anexado a um gráfico antes do encerramento do terminal);
- carga de um template (se o Expert Advisor anexado a um gráfico está especificado no template);
- alteração de um perfil (se o Expert Advisor está anexado a um dos gráficos do perfil);
- conexão a uma conta, mesmo se o número da conta for o mesmo (se o Expert Advisor estava anexado ao gráfico antes da autorização do terminal no servidor.

Expert Advisors são descarregados nos seguintes casos:

- ao desanexar um Expert Advisor de um gráfico;
- Se um novo Expert Advisor é anexo a um gráfico, e um outro Expert Advisor já estava anexado, este Expert Advisor é descarregado.
- encerramento do terminal (se o Expert Advisor estiver anexado a um gráfico);
- carga de um template, se um Expert Advisor está anexado a um gráfico;
- fechamento de um gráfico, no qual o Expert Advisor está anexado.
- alteração de um perfil, se o Expert Advisor está anexado a um dos gráficos do perfil alterado;

© 2000-2024, MetaQuotes Ltd.

- alteração da conta na qual o terminal está conectado (se o Expert Advisor estava anexado a um gráfico antes da autorização do terminal no servidor;
- calling the ExpertRemove() function.

Em caso de alteração de ativo ou período de um gráfico, no qual o Expert Advisor está anexado, Expert Advisors não são carregados ou descarregados. Neste caso o terminal cliente subseqüentemente chama os handlers OnDeinit() sobre o ativo/período antigo e OnInit() sobre o novo ativo/período; valores de variáveis globais e variáveis estáticas não são redefinidos. Todos os eventos que foram recebidos pelo Expert Advisor antes da inicialização ser concluída (função OnInit()) são pulados.

Carga e Descarga de Scripts

Scripts são carregados imediatamente após eles serem anexados a um gráfico e descarregados imediatamente após eles concluírem sua operação. OnInit() e OnDeinit() não são chamados por scripts.

Quando um programa é descarregado (excluído de um gráfico) o terminal cliente realiza a desinicialização de variáveis globais e excluí a fila de eventos. Neste caso a desinicialização significa a redefinição de todas as variáveis de tipo string, desalocação de objetos de array dinâmico e chamada de seus destrutores se eles estiverem disponíveis.

Carga e Descarga de Serviços

Os serviços são carregados imediatamente após o início do terminal, se, ao parar o terminal, eles são iniciados. Os serviços são descarregados imediatamente após o término de seu trabalho.

Os serviços têm um único manipulador OnStart() no qual você pode organizar um loop infinito de recebimento e de processamento de dados, por exemplo, ao criar e atualizar símbolos personalizados usando funções de rede.

Ao contrário de EAs, de indicadores e de scripts, os serviços não estão vinculados a um gráfico específico, portanto, para iniciar o serviço é fornecido um mecanismo separado. A criação de uma nova instância do serviço é feita a partir do Navegador usando o comando Adicionar serviço. Para iniciar, parar, remover uma instância de serviço, use seu menu. Para gerenciar todas as instâncias, use o menu do próprio serviço.

Para um melhor entendimento do funcionamento de um Expert Advisor, nós recomendamos compilar o código do seguinte Expert Advisor e realizar ações de carga/descarga, alteração de template, alteração de ativo, alteração de período, etc:

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass construtor"); }
   ~CTestClass() { Print("CTestClass destruidor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Inicialização");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Desinicialização com motivo",reason);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

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

Também Veja

Eventos do terminal cliente, Handlers de Evento

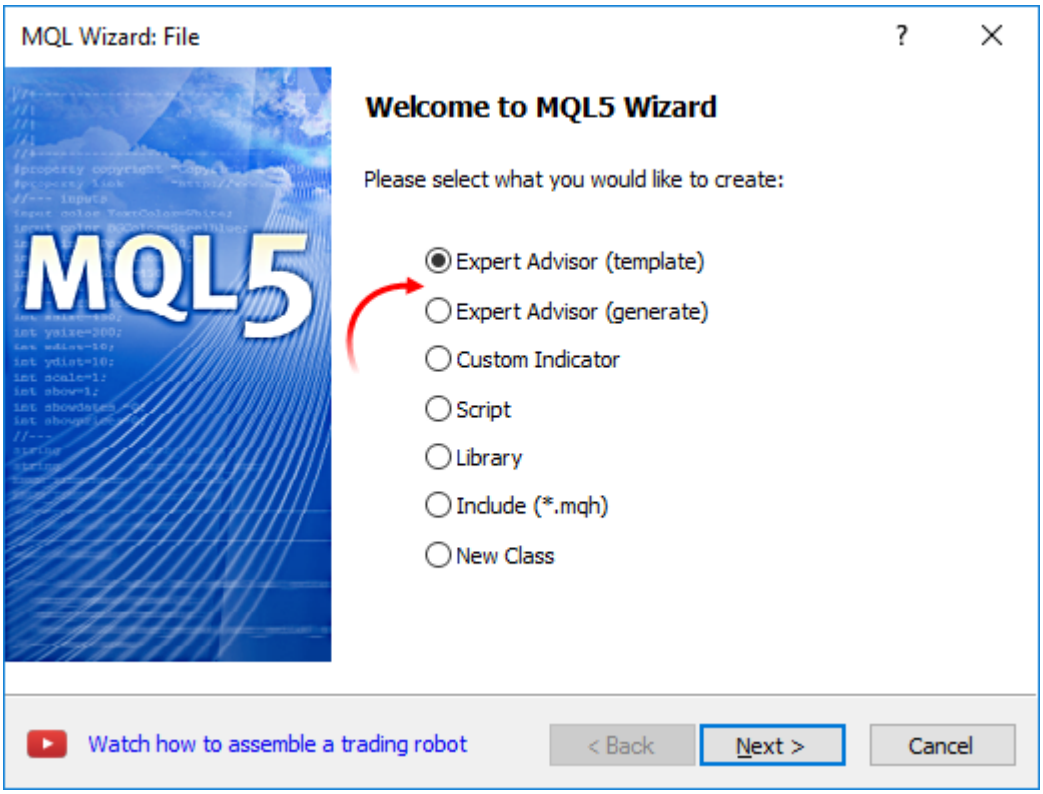
© 2000-2024, MetaQuotes Ltd.

Trade Permission

Trade Automation

A linguagem MQL5 fornece um grupo especial de funções de negociação projetado para o desenvolvimento de sistemas de negociação automatizados. Programas desenvolvidos para negociação automatizada sem intervenção humana são chamados Expert Advisors ou robôs de negociação. Para criar um Expert Advisor no MetaEditor, abra o MQL5 Wizard e selecione uma das duas opções:

- Expert Advisor (modelo) – permite-lhe criar um modelo com uma função de manipulação de eventos que devem ser complementados com toda a funcionalidade necessária por meio de programação.
- Expert Advisor (generate) – permite você desenvolver um robô de negociação completoSimplesmente selecionando os módulos necessários: módulo de negociação de sinais, módulo de gerenciamento de dinheiro e módulo de parada final.



As funções de negociação podem funcionar apenas em Expert Advisors e scripts. A negociação não é permitida para os indicadores.

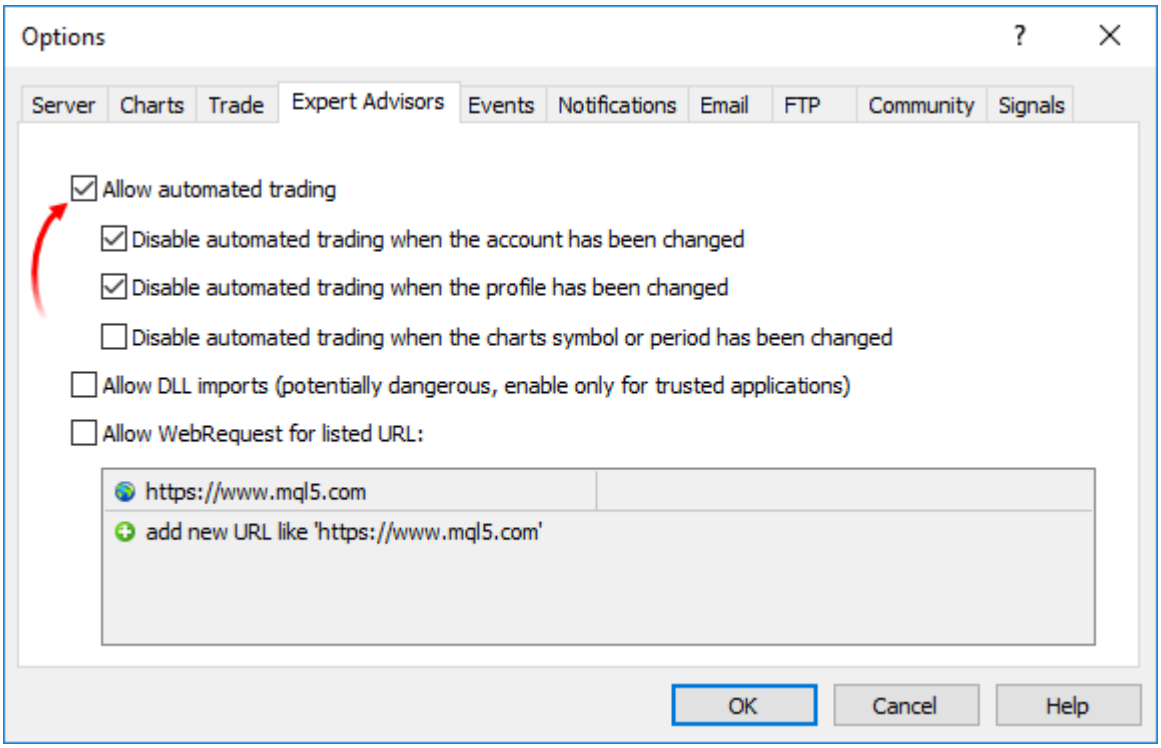
Verificando a Permissão para Realizar uma Negociação Automatizada

A fim de desenvolver um Expert Advisor confiável capaz de trabalhar sem intervenção humana, é necessário organizar um conjunto de verificações importantes. Primeiro, devemos verificar programaticamente se a negociação é permitida em tudo. Esta é uma verificação básica que é indispensável ao desenvolver qualquer sistema automatizado.

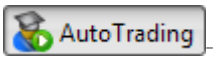
Verificação da permissão para realizar a negociação automatizada no terminal

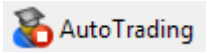
© 2000-2024, MetaQuotes Ltd.

As configurações de terminal fornecem a você a capacidade de permitir ou proibir a negociação automatizada de todos os programas.



Você pode alternar a opção de negociação automatizada diretamente no painel Padrão do terminal:

- 

- negociação automatizada habilitada, funções de negociação em aplicativos lançados são permitidos para uso.
- 

- negociação automatizada desabilitada, as aplicações em execução não conseguem executar funções de negociação.

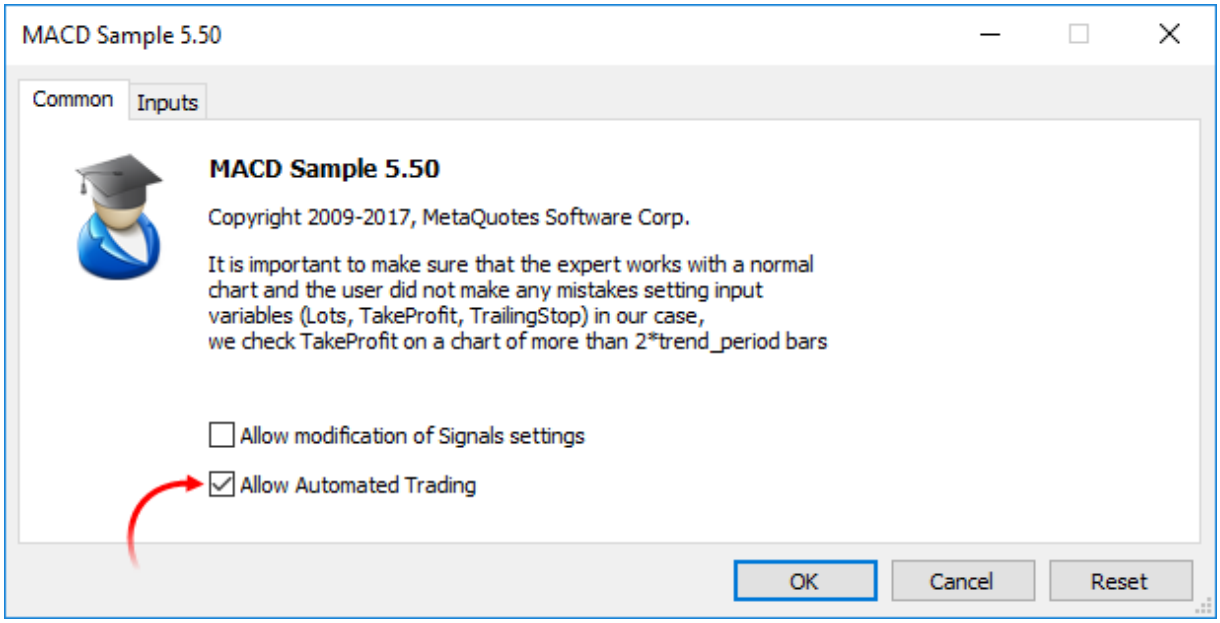
Verificar amostra:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Verifique se a negociação automatizada é permitida nas configurações do terr
```

Verificando se a negociação é permitida para uma determinada execução do Expert Advisor/script

Você pode permitir ou proibir a negociação automatizada para um determinado programa ao lançá-lo. Para fazer isso, use a caixa de seleção especial nas propriedades do programa.

© 2000-2024, MetaQuotes Ltd.



Verificar amostra:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Verifique se a negociação automatizada é permitida nas configurações do t
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("A negociação automatizada é proibida nas configurações do ",__FILE__);
  }
```

Verificando se a negociação é permitida para qualquer Expert Advisors/scripts para a conta corrente

A negociação automatizada pode ser desativada no lado do servidor de negociação. Verificar amostra:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert("Negociação automatizada é proibida para a conta ",AccountInfoInteger(ACCO
   " no lado do servidor de negociação");
```

Se a negociação automatizada estiver desativada para uma conta de negociação, as operações de negociação de Expert Advisors/scripts não serão executadas.

Verificando se a negociação é permitida para a conta corrente

Em alguns casos, quaisquer operações de negociação são desativadas para uma determinada conta de negociação - nem manual nem automatizado de negociação pode ser realizada. Exemplo de verificação quando uma senha de investidor foi usada para conectar-se a uma conta de negociação:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("A negociação é proibida para a conta ",AccountInfoInteger(ACCOUNT_LOGIN
         ".\n Talvez uma senha de investidor tenha sido usada para conectar-se à co
         "\n Verifique o diário do terminal para a seguinte entrada:",
```

© 2000-2024, MetaQuotes Ltd.

```
                "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': negociação foi desativada -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) deve retornar **falsa** nos casos seguintes:

- Sem conexão com o servidor de negociação. Isso pode ser verificado usando TerminalInfoInteger(TERMINAL_CONNECTED);
- Conta de negociação comutada para modo somente leitura (enviada para o arquivo);
- A negociação na conta é desativada no lado do servidor de negociação;
- A conexão com uma conta de negociação foi realizada no modo Investidor.

Veja também

Propriedades do terminal do cliente, Propriedades da Conta, Propriedades de um programa MQL5 em execução

© 2000-2024, MetaQuotes Ltd.

Eventos do Terminal Cliente

Init

Imediatamente após o terminal cliente carregar um programa (uma Expert Advisor ou um indicador customizado) e começar o processo de inicialização de variáveis globais, o evento Init será enviado, e será processado pelo handler de evento OnInit(), caso exista. Este evento também é gerado após um instrumento financeiro e / ou prazo gráfico ser alterado, depois de um programa ser recompilado no MetaEditor, depois dos parâmetros de entrada serem alterados a partir da janela de configuração de um Expert Advisor ou um indicador personalizado. Um Expert Advisor é também inicializado após a conta ser alterada. O evento Init não é gerado para scripts.

Deinit

Antes de variáveis globais serem desinicializadas e o programa (Expert Advisor ou indicador customizado) ser descarregado, o terminal cliente envia o evento Deinit para o programa. Deinit é também gerado quando o terminal cliente é encerrado, quando um gráfico é fechado, logo antes o ativo e/ou período ser alterado, em uma recompilação de programa bem sucedida, quando parâmetros de input são alterados e quando a conta é alterada.

O motivo de desinicialização pode ser obtido a partir do parâmetro passado para a função OnDeinit(). A execução da função OnDeinit() está restrita a 2.5 segundos. Se durante este tempo a função não for concluída, então ela é finalizada de forma forçada. O evento Deinit (Desinicializar) não é gerada para scripts.

Start

Evento Start – evento especial para executar um script ou serviço depois que ele é carregado. Este evento é tratado pela função OnStart. Evento Start não é enviado para EAs e indicadores personalizados.

NewTick

O evento NewTick é gerado caso existam novas cotações, ele é processado pelo OnTick() de Expert Advisors anexados. No caso de uma nova cotação ser recebida enquanto a função OnTick estiver processando a cotação anterior, a nova cotação será ignorada pelo Expert Advisor, porque o evento correspondente não será enfileirado.

Todas as novas cotações que são recebidas enquanto o programa está rodando são ignoradas até que a execução da função OnTick() esteja concluída. Após isso, a função rodará somente após uma nova cotação ser recebida. O evento NewTick é gerado independentemente da negociação automática ser permitida ou não (botão "AutoTrading Permite/Proíbe"). A proibição de negociação automática significa somente que o envio de solicitações de negociação a partir de um Expert Advisor não é permitido, enquanto o Expert Advisor continua trabalhando.

A proibição de negociação automática ao apertar o botão apropriado não parará a execução corrente da função OnTick().

Calculate

O evento Calculate é gerado somente para indicadores logo após o evento Init ser enviado e em qualquer alteração de dados de preços. Ele é processado pela função OnCalculate.

© 2000-2024, MetaQuotes Ltd.

Timer

O evento Timer é periodicamente gerado pelo terminal cliente para o Expert Advisor que ativou o timer através da função EventSetTimer. Geralmente, esta função é chamada na função OnInit. O processamento do evento Timer é realizado pela função OnTimer. Após a operação do Expert Advisor estar concluída, é necessário destruir o timer usando a função EventKillTimer, que é geralmente chamada na função OnDeinit.

Trade

O evento Trade é gerado quando uma operação de trade é concluída em um servidor de negociação. O evento Trade é tratado pela função OnTrade() para as seguintes operações de negociação:

- envio, modificação e remoção de uma ordem pendente;
- cancelamento de uma ordem pendente com ausência de dinheiro suficiente ou expiração;
- ativação de uma ordem pendente;
- abertura, acréscimo ou encerramento de uma posição (ou parte da posição);
- modificação da posição aberta (alteração de stops - Stop Loss e/ou Take Profit).

TradeTransaction

Ao realizar algumas ações específicas em uma conta de negociação, seu estado muda. Tais ações includem:

- Envio de uma solicitação de negociação a partir de algum aplicativo MQL5 no terminal usando as funções OrderSend e OrderSendAsync e sua posterior execução.
- Envio de uma solicitação de negociação por meio da interface gráfica do terminal e sua posterior execução.
- Ativação de ordens pendentes e ordens de stop no servidor.
- Realização de operações no lado de um servidor de negociação.

As seguintes transações de negociação são realizadas como resultado destas ações:

- tratamento de uma solicitação de negociação
- mudança de ordens de abertura
- mudança de histórico de ordens
- mudança de histórico de operações (deals)
- mudança de posições

Por exemplo, ao enviar uma ordem de compra de mercado, ela é tratada, uma ordem de compra apropriada é criada para a conta, a ordem é então executada e removida da lista de ordens em aberto, e então ela é adicionada ao histórico de ordens, uma apropriada operação (deal) é adicionada ao histórico e uma nova posição é criada. Todas estas ações são transações de negociação. A chegada de tal transação no terminal é um evento TradeTransaction. Este evento é tratado pela função OnTradeTransaction.

Tester

O evento Tester é gerado após um teste de um Expert Advisor sob dados de histórico ter acabado. O evento é tratado pela função OnTester().

© 2000-2024, MetaQuotes Ltd.

TesterInit

O evento TesterInit é gerado no início de uma otimização no Provador de Estratégia antes do primeiro passo de otimização. O evento TesterInit é tratado pela função OnTesterInit().

TesterPass

O evento TesterPass é gerado quando uma nova estrutura de dados é recebido. O evento TesterPass é tratado pela função OnTesterPass().

TesterDeinit

O evento TesterDeinit é gerado após o fim de um otimização de um Expert Advisor no Provador de Estratégia. O evento TesterDeinit é tratado pela função OnTesterDeinit().

ChartEvent

O evento ChartEvent é gerado pelo terminal cliente quando um usuário está trabalhando com um gráfico:

- pressionar de teclado, quando a janela de gráfico está com foco
- objeto gráfico criado
- objeto gráfico excluído
- pressionar de mouse sobre um objeto gráfico do gráfico
- movimentação do objeto gráfico usando o mouse
- fim de edição de texto no LabelEdit

Também existe o evento customizado ChartEvent, que pode ser enviado para um Expert Advisor através de qualquer programa mql5 por meio do uso da função EventChartCustom. O evento é processado pela função OnChartEvent.

BookEvent

O evento BookEvent é gerado pelo terminal cliente após a Profundida de Mercado ser alterada; ele é processado pela função OnBookEvent. Para começar a geração de BookEvent para um ativo especificado, é necessário subscrever o ativo para este evento através do uso da função MarketBookAdd.

Para desfazer a subscrição de BookEvent para um ativo especificado, é necessário chamar a função MarketBookRelease. O evento BookEvent é um evento do tipo-transmissão - isso significa que é suficiente inscrever apenas um Consultor Especializado (Expert Advisor) para este evento, e todos os outros Consultores Especializados (Expert Advisors) que têm o manipulador de eventos OnBookEvent, vão recebê-lo. É por isso que é necessário analisar o nome do ativo, que é passado para um handler como um parâmetro.

Também Veja

Handlers de Evento, Execução de Programa

© 2000-2024, MetaQuotes Ltd.

Recursos

Usando gráficos e som em programas MQL5

Programas em MQL5 permitem trabalhar com som e arquivos gráficos:

- PlaySound() executa um arquivo de som.
- ObjectCreate() permite criar interfaces de usuário usando os objetos gráficos OBJ_BITMAP e OBJ_BITMAP_LABEL.

PlaySound()

Exemplo de chamada da função PlaySound():

```
//+------------------------------------------------------------------+
//| A função chama OrderSend() padrão e toca um som                  |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- envia uma solicitação para um servidor
   OrderSend(request,result);
   //--- se a solicitação for aceite, executa o som Ok.wav
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- se falhar, toca um alarme a partir do arquivo timeout.wav
   else PlaySound("timeout.wav");
  }
```

O exemplo mostra como tocar sons a partir de arquivos 'Ok.wav' e 'timeout.wav', que estão incluídos no pacote do terminal padrão. Estes arquivos estão localizados na pasta **terminal_directory\Sounds**. Aqui, **terminal_directory** é uma pasta, a partir do qual o terminal de cliente da MetaTrader 5 é iniciado. A localização do diretório do terminal pode ser encontrado a partir de um programa MQL5 da seguinte forma:

```
//--- Pasta, na qual dados de terminal são armazenados
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Você pode usar arquivos de som não somente da pasta **terminal_directory\Sounds**, mas também de qualquer sub-pasta localizada em **terminal_data_directory\MQL5.** Você pode descobrir a localização do diretório de dados do terminal a partir do menu do terminal "Arquivo" -> "Abrir" dados do terminal ou usar um método de programa:

```
//--- Pasta, na qual dados de terminal são armazenados
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Por exemplo, se o arquivo de som 'Demo.wav' está localizado em terminal_data_directory\MQL5\Files, então a chamada de PlaySound() deveria ser escrita da seguinte forma:

```
//--- toca Demo.wav a partir da pasta terminal_directory_data\MQL5\Files\Demo.wav
   PlaySound("\\Files\\Demo.wav");
```

© 2000-2024, MetaQuotes Ltd.

Observe que no comentário o caminho do arquivo está escrito usando uma barra invertida "\", e na função é usado "\\".

Ao se especificar o caminho, sempre use barras invertidas duplas como separador, porque uma única barra invertida é um controle de símbolo para o compilador ao lidar com constantes de cadeias de caracteres e constantes de caracteres no código fonte do programa.

Call PlaySound() function with NULL parameter to stop playback:

```
//--- Chamar o PlaySound() com parâmetro NULL interrompe a reprodução
   PlaySound(NULL);
```

ObjectCreate()

Exemplo de um Expert Advisor, que cria um rótulo gráfico (OBJ_BITMAP_LABEL) usando a função ObjectCreate().

```
string label_name="currency_label";        // nome do objeto OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";    // caminho do arquivo terminal_dara_directo
string dollar    ="\\Images\\dollar.bmp";  // caminho do arquivo terminal_dara_directo
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- cria um botão OBJ_BITMAP_LABEL, se ele já não tiver sido criado
   if(ObjectFind(0,label_name)<0)
     {
      //--- tenta criar objeto OBJ_BITMAP_LABEL
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- vincula o botão ao canto superior esquerdo do gráfico
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- agora configura as propriedades do objeto
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- redefine o código do último erro para 0
         ResetLastError();
         //--- carrega uma figura para indicador o estado "Pressionado" do botão
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- testa o resultado
         if(!set)
           {
            PrintFormat("Falha no download a partir do arquivo de imagem %s. Código de
           }
         ResetLastError();
         //--- carrega uma figura para indicador do estado "Não pressionado" do botão
```

```
          set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

          if(!set)
            {
             PrintFormat("Falha no download a partir do arquivo de imagem %s. Código de
            }
          //--- envia um comando para um gráfico para reatualização, a fim de que o bot
          ChartRedraw(0);
         }
       else
         {
          //--- falha ao criar um objeto, notificar
          PrintFormat("Falha ao criar objeto OBJ_BITMAP_LABEL. Error code %d",GetLastEr
         }
      }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- exclui um objeto de um gráfico
   ObjectDelete(0,label_name);
  }
```

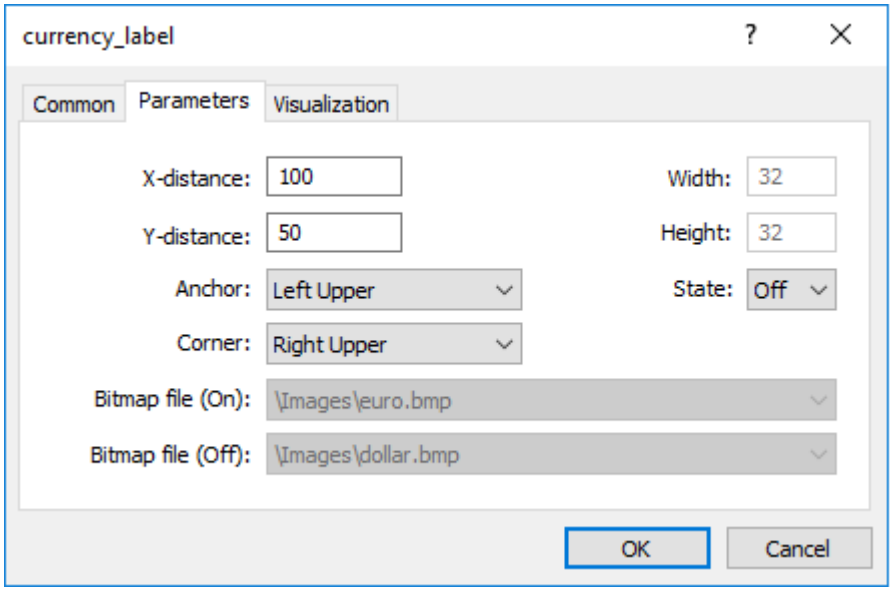
A criação e configuração do objeto gráfico de nome currency_label são executados na função OnInit(). Os caminhos para os arquivos gráficos são definidos nas variáveis globais **euro** e **dollar**, uma barra invertida dupla é usada como separador:

```
string euro      ="\\Images\\euro.bmp";     // caminho do arquivo terminal_dara_directo
string dollar    ="\\Images\\dollar.bmp";   // caminho do arquivo terminal_dara_directo
```

Os arquivos estão localizados na pasta **terminal_data_directory**\MQL5\Images.

O Objeto OBJ_BITMAP_LABEL é na realizada um botão, que exibe uma das duas imagens, dependendo do estado do botão (pressionado ou não pressionado): euro.bmp ou dollar.bmp.

© 2000-2024, MetaQuotes Ltd.



O tamanho do botão com uma interface gráfico é automaticamente ajustada para o tamanho da figura. A imagem é alterada por um clique do botão esquerdo do mouse sob o objeto OBJ_BITMAP_LABEL (**"Disable selection" deve estar definida nas propriedades**). O objeto OBJ_BITMAP é criado da mesma forma - ele é usado para criar o pano de fundo com um imagem necessária.

O valor da propriedade OBJPROP_BMPFILE, que é responsável pela aparência dos objetos OBJ_BITMAP e OBJ_BITMAP_LABEL, e pode ser alterado dinamicamente. Isso permite criar várias interfaces interativas de usuário para programas MQL5.

Inclusão de recursos em arquivos executáveis durante compilação de programas MQL5

Um programa mql5 pode precisar de um monte de diferentes recursos baixáveis na forma de arquivos de imagem e som. A fim de eliminar a necessidade de transferir todos estes arquivos ao mover um arquivo executável em MQL5, a diretriz do compilador #resource deve ser usada:

```
#resource path_to_resource_file
```

O comando #resource diz ao compilador que o recurso no caminho especificado **path_to_resource_file** deve ser incluído dentro do arquivo executável EX5. Assim, todos os sons e imagens necessários podem ser alocados diretamente dentro de um arquivo EX5, de modo que não exista necessidade de transferir separadamente os arquivos nele usados, caso você queira executar o programa em um terminal diferente. Qualquer arquivo EX5 pode conter recursos, e qualquer programa EX5 pode usar recursos de um outro programa EX5.

Os arquivo em formato BMP e WAV são automaticamente comprimidos antes de serem incluídos em um arquivo EX5. Isso significa que além de criar programas completos em MQL5, o uso de recursos também permite reduzir o tamanho total dos arquivos necessários ao usar gráficos e sons, quando comparado com a forma usual de escrever um programa MQL5.

O arquivo de recurso não deve exceder 16 Mb.

© 2000-2024, MetaQuotes Ltd.

Busca por recursos específicos pelo compilador

Um recurso é inserido usando o comando #resorce "<path to the resource file>"

```
#resource "<path_to_resource_file>"
```

O comprimento da constante string <path_to_resource_file> não de exceder 63 caracteres.

O compilador busca por um recurso no caminho especificado na seguinte ordem:

- se o separador barra invertida "\" (escrito como "\\") é colocado no começo do caminho, ele busca pelo recurso referente ao diretório **terminal_data_directory**\MQL5\,
- se não houver barra invertida, ele busca pelo recurso referente à localização do arquivo de recurso, na qual o recurso foi escrito.

O caminho do recurso não pode conter as sub-cadeias "..\\" e ":\\".

Exemplos de inclusão de recurso:

```
//--- correct specification of resources
#resource "\\Images\\euro.bmp" // euro.bmp está localizado em terminal_data_directory\
#resource "picture.bmp"        // picture.bmp está localizado no mesmo diretoria que c
#resource "Resource\\map.bmp"  // O recurso está localizado em source_file_directory\F

//--- especificação incorreta de recursos
#resource ":picture_2.bmp"     // não deve conter ":"
#resource "..\\picture_3.bmp"  // não deve conter ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //mais
```

Uso de Recursos

Nome de recurso

Após um recurso ser declarado usando a diretiva #resource, ele pode ser usado em qualquer parte de um programa. O nome do recurso é seu caminho sem uma barra invertida no começo da linha, que define o caminho do recurso. Para usar seu próprio recurso no código, o sinal especial "::" deve ser adicionado antes do nome do recurso.

Exemplos:

```
//--- exemplos de especificação de recurso e seus nomes em comentários
#resource "\\Images\\euro.bmp"          // nome do recurso - Images\euro.bmp
#resource "picture.bmp"                 // nome do recurso - picture.bmp
#resource "Resource\\map.bmp"           // nome do recurso - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // nome do recurso - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";          // nome do recurso - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // nome do recurso - Sounds\thrill.wav"
...
```

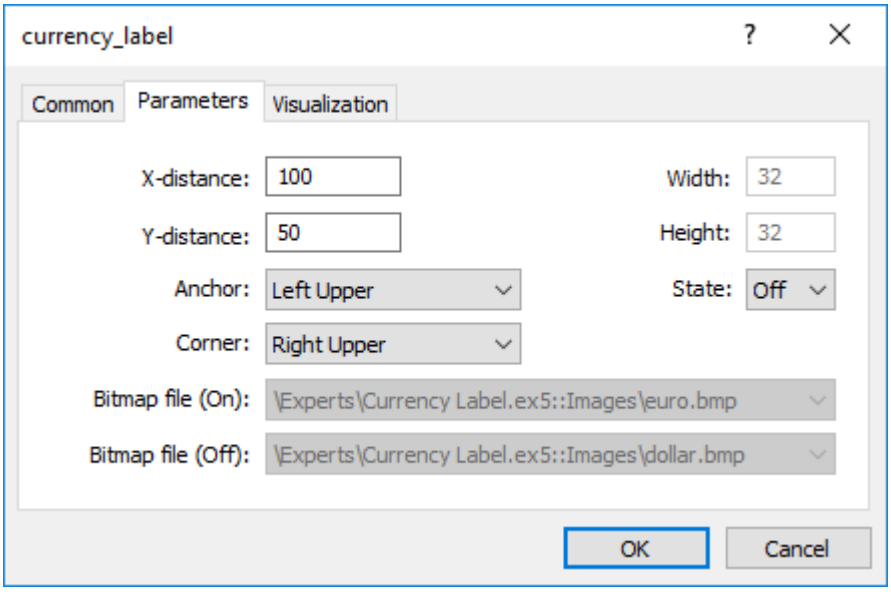
© 2000-2024, MetaQuotes Ltd.

```
//--- utilização dos recursos
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Deve se notar que ao definir imagens a partir de um recurso para os objetos OBJ_BITMAP e OBJ_BITMAP_LABEL, o valor da propriedade OBJPROP_BMPFILE não pode ser modificado manualmente. Por exemplo, para criar OBJ_BITMAP_LABEL nós usados euro.bmp e dollar.bmp.

```
#resource "\\Images\\euro.bmp";    // euro.bmp está localizado em terminal_data_direct
#resource "\\Images\\dollar.bmp";  // dollar.bmp está localizado em terminal_data_dire
```

Ao visualizar as propriedades deste objeto, veremos que as propriedades BitMap File (On) e BitMap File (Off) são esmaecidas e não podem ser alteradas manualmente:



Usando os recursos em outros programas MQL5

Existe uma outra vantagem no uso de recurso - em qualquer programa MQL5, os recursos de um outro arquivo EX5 podem ser usados. Assim, os recursos de um arquivo EX5 podem ser usados em muitos outros programas MQL5.

A fim de usar um nome de recurso de um outro arquivo, ele deve ser especificado como <path_EX5_file_name>::<resource_name>. Por exemplo, suponha que o script Draw_Triangles_Script.mq5 contém um recurso para uma imagem no arquivo triangle.bmp:

```
#resource "\\Files\\triangle.bmp"
```

© 2000-2024, MetaQuotes Ltd.

Então seu nome, para uso no script em si, se assemelhará a "Files\triangle.bmp", e a fim de usá-lo, "::" deve ser adicionado ao nome do recurso.

```
//--- usando o curso no script
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

A fim de usar o mesmo recurso a partir de um outro programa, por exemplo, a partir de um Expert Advisor, precisamos adicionar ao nome do recurso o caminho para o arquivo EX5 correspondente ao **terminal_data_directory\MQL5** e o nome arquivo EX5 do script - **Draw_Triangles_Script.ex5**. Suponha que o script está localizado na pasta padrão **terminal_data_directory\MQL5**Scripts\, então o chamado deve ser escrito da seguinte forma:

```
//--- usando um recurso de um script em um EA
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Se o caminho para o arquivo executável não for especificado ao chamar o recurso de um outro EX5, o arquivo executável é procurado na mesma pasta que contém o programa que chama o recurso. Isso significa que se um Expert Advisor chamar um recurso de Draw_Triangles_Script.ex5 sem especificar o caminho, como abaixo:

```
//--- chama recurso de script em um EA sem especificar o caminho
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

então o arquivo será procurado na pasta **terminal_data_directory\MQL5**Experts\, caso o Expert Advisor esteja localizado em **terminal_data_directory\MQL5**Experts\.

Trabalhando com indicadores personalizados que estão conectados como recursos

O funcionamento de programas MQL5 pode exigir um ou mais indicadores personalizados, eles podem ser incluídos no código do programa executável MQL5. A inclusão de indicadores como recursos simplifica a distribuição de programas.

Exemplo de conexão e utilização do indicador personalizado SampleIndicator.ex5, localizado na pasta: **diretório_de_dados_do_terminal**\MQL5\Indicators\:

```
//+------------------------------------------------------------------+
//|                                                   SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
```

```
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
//---
   return(INIT_SUCCEEDED);
  }
```

Os casos em que o indicador personalizado, na função OnInit(), cria uma ou mais cópias de si mesmo exigem uma análise separada. Lembre-se que para usar um recurso a partir de um programa mql5, é preciso especificá-lo no formato: **<caminho_do_nome_do_arquivo_EX5>::<nome_do_recurso>**.

Por exemplo, se o indicador SampleIndicator.ex5 estiver incluído no Expert Advisor SampleEA.ex5 como um recurso, então o caminho para si mesmo será especificado ao chamar iCustom() na função de inicialização do indicador personalizado, isso será da seguinte forma: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". Se for definido explicitamente o caminho, o indicador personalizado SampleIndicator.ex5 será firmemente ligado ao Advisor SampleEA.ex5 e perderá a capacidade de trabalhar de forma independente.

O caminho para si mesmo pode ser obtido utilizando a função GetRelativeProgramPath(), a seguir, um exemplo de utilização:

```
//+------------------------------------------------------------------+
//|                                              SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- maneira errada para especificar uma referência para si próprio
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- maneira correta para obter uma referência a si mesmo
  string path=GetRelativeProgramPath();
//--- indicator buffers mapping
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Indicator: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Indicator handle=",handle);
//---
   return(INIT_SUCCEEDED);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- obtemos o caminho absoluto para o programa
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- encontramos a posição da sub-cadeia de caracteres "\MQL5\"
   int    pos =StringFind(path,"\\MQL5\\");
//--- sub-cadeia não encontrada - erro
   if(pos<0)
      return(NULL);
//--- ignoramos o diretório "\MQL5"
   pos+=5;
//--- ignoramos '\' extras
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- se se tratar de um recurso, retornamos o caminho em relação ao diretório MQL5
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- encontramos o delimitador para o primeiro sub-diretório em MQL5 (por exemplo, MQ
//--- se ele não existir, retornamos o caminho em relação ao diretório MQL5
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- retornamos o caminho em relação ao sub-diretório (например, MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- Valor de retorno do prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Variáveis de recurso

Os recursos podem ser declarados usando as variáveis de recurso, e tratá-los como se fossem uma variável do tipo apropriado. Formato do declaração:

```
#resource caminho_para_o_arquivo_do_recurso as tipo_de_variável_de_recurso nome_de_var
```

© 2000-2024, MetaQuotes Ltd.

Exemplos de declaração:

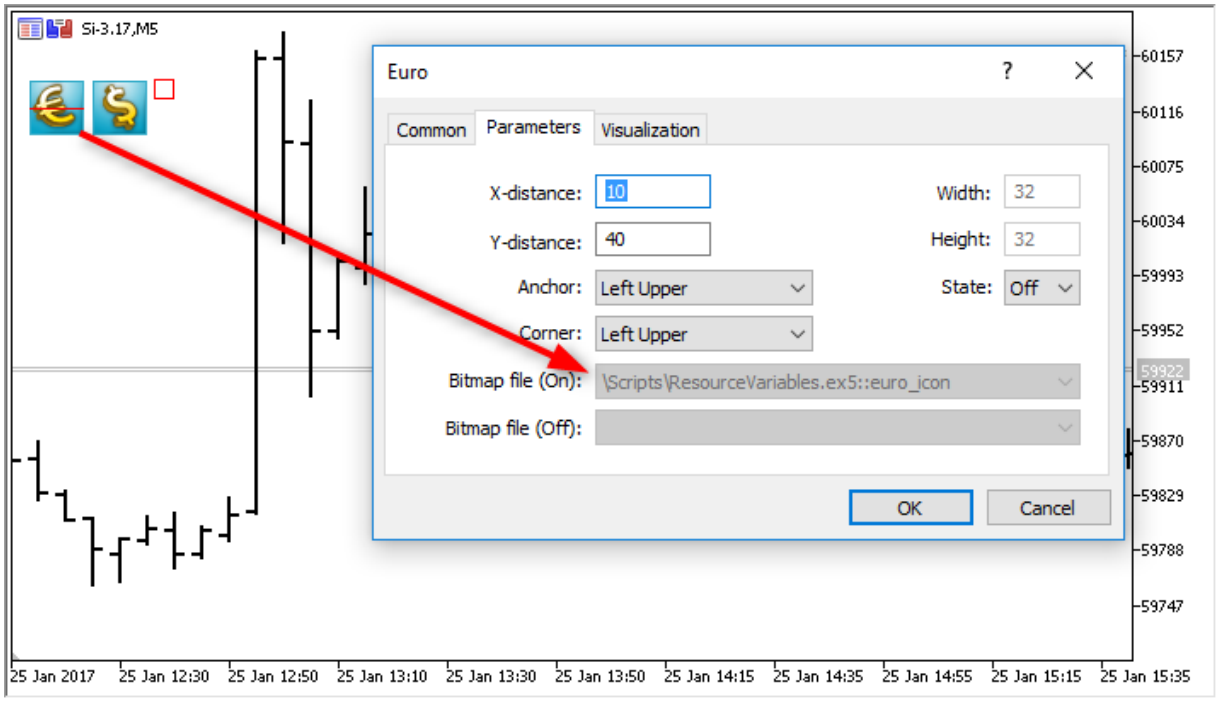
```
#resource "data.bin" as int ExtData[]              // declaração de matriz de tipo nume
#resource "data.bin" as MqlRates ExtData[]         // declaração de matriz de estrutura
//--- cadeias de caracteres
#resource "data.txt" as string ExtCode             // declaração de cadeias de caracte
//--- recursos gráficos
#resource "image.bmp" as bitmap ExtBitmap[]        // declaração de matriz unidimensio
#resource "image.bmp" as bitmap ExtBitmap2[][]     // declaração de matriz bidimensiona
```

Ao utilizar tal declaração, os dados deste recurso podem ser tratados por intermédio de uma variável, o endereçamento automático via "::<rsource name>" não funciona.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  Função de criação de objeto OBJ_BITMAP_LABEL usando o recurso|
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- imprimimos os tamanhos da imagem [width, height], que é armazenada na variável c
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- alteramos a imagem no euro - desenhamos uma faixa horizontal no meio
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- criamos o recurso gráfico usando a variável de recurso
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- criamos objeto do rótulo gráfico Euro, para o qual colocamos a imagem a partir d
   Image("Euro","::euro_icon",10,40);
//--- outro modo de utilizar o recurso, nós não podemos desenhar nele
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- a maneira direta de endereçamento para o recurso euro.bmp não está disponível, u
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // acontece o erro de
  }
```

Resultado de execução do script, estão criados apenas dois objetos OBJ_BITMAP_LABEL de três. Ao mesmo tempo, na imagem do primeiro objeto, nós vemos uma linha vermelha no meio.

© 2000-2024, MetaQuotes Ltd.



Uma vantagem importante do uso de recursos é que os arquivos de recurso - antes de serem incluídos num arquivo EX5 executável e compilados - são comprimidos automaticamente. Assim, o uso de variáveis de recurso permite não só empacotar diretamente os dados necessários num arquivo EX5 executável, mas também reduzir o número e o tamanho total dos arquivos em comparação com o método convencional de escrita de programas MQL5.

A utilização de variáveis de recurso é particularmente útil para publicar produtos no Mercado.

Características

- O tipo especial de variável de recurso *bitmap* informa ao compilador que o recurso é uma representação gráfica. Essas variáveis recebem o tipo uint.
- A matriz-variável de recurso de tipo *bitmap* pode ter duas dimensões, neste caso, o tamanho da matriz será definido como [altura_de_imagem][largura_de_imagem]. No caso de uma matriz unidimensional, o número de elementos será definido como o produto de altura_de_imagem*largura_de_imagem.
- Ao carregar imagens de 24 bits, para todos os pixels da imagem de componente de canal-alfa, define-se como 255.
- Ao carregar imagens de 32 bits, para todos os pixels da imagem de componente de canal-alfa, define-se como 255.
- Após carregar uma imagem de 32 bits com canal-alfa não acontece nenhuma manipulação de pixels.
- O tamanho do arquivo de recurso não pode ser maior do que 128 MB.
- Para arquivos de sequência de caracteres, a codificação de BOM (cabeçalho) é detectada automaticamente. Se não houver nenhum BOM, a codificação será determinada pelo conteúdo. São suportados arquivos codificados em ANSI, UTF-8 e UTF-16. Todas as cadeias de caracteres são convertidas para Unicode.

Programas em OpenCL

© 2000-2024, MetaQuotes Ltd.

A utilização de variáveis - de sequências de caracteres - de recurso pode facilitar muito a escrita de alguns programas. Por exemplo, você pode escrever o código de um programa OpenCL num arquivo CL separado e, em seguida, incluir esse arquivo - como uma cadeia de caracteres - nos recursos de seu programa MQL5.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- executa ações futuras com o programa OpenCL
  }
```

Neste exemplo, sem a utilização de uma variável de recurso*cl_program*, você teria de descrever esse código como uma variável de cadeia grande.

Veja Também

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Funções de Arquivo

© 2000-2024, MetaQuotes Ltd.

Chamadas de Funções Importadas

Para importar funções durante a execução de um programa-mql5, o terminal cliente usar "early binding". Isso significa que se um programa tem uma chamada de uma função importada, o correspondente módulo (ex5 ou dll) é carregada durante a carga do programa. MQL5 e bibliotecas DLL são executadas na thread do módulo da chamada.

Não é recomendado usar o nome do módulo com especificação completa para ser carregado, como *Drive:\Directory\FileName.Ext*. As bibliotecas MQL5 são carregadas a partir da pasta *terminal_dir\MQL5\Libraries*. Se a biblioteca não for encontrada, então o terminal tenta carregá-la a partir da pasta *terminal_dir\experts*.

As bibliotecas de sistema (DLL) são carregadas pelas regras do sistema operacional. Se a biblioteca já estiver carregada (por exemplo, um outro Expert Advisor, e mesmo a partir de um outro terminal cliente, executando em paralelo, então ele faz solicitações para a biblioteca já carregada. Caso contrário, ele realiza uma busca na seguinte seqüência:

1. Diretório do qual o módulo que importa a dll foi iniciado. O módulo aqui é um Expert Advisor, um script, um indicador ou uma biblioteca EX5.
2. Diretório terminal_data_directory\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries).
3. Diretório do qual o terminal cliente MetaTrader 5 foi iniciado.
4. Diretório de sistema.
5. Diretório do Windows.
6. Diretório corrente.
7. Diretórios listados na variável de sistema PATH.

Se uma biblioteca DLL usa uma outra DLL no seu funcionamento, a primeira DLL não pode ser carregada no caso de não existir a segunda DLL.

Antes de um Expert Advisor (script, indicador) ser carregado, uma lista comum de todos os módulos de biblioteca EX5 é formada. Vai ser usado a partir de um Expert Advisor carregado(script, indicador)e das bibliotecas desta lista. Assim, é necessário apenas uma única carga de módulos de bibliotecas EX5 usadas muitas vezes. Bibliotecas usam variáveis predefinidas do Expert Advisor (script, indicador) que as chamam.

A biblioteca importada EX5 é procurada na seguinte seqüência:

1. Diretório, caminho correspondente ao diretório do Expert Advisor (script, indicador) que importa o EX5).
2. Diretório terminal_directory\MQL5\Libraries.
3. Diretório MQL5\Libraries no diretório comum a todos os terminais clientes MetaTrader 5 (Common\MQL5\Libraries).

Funções importadas DLL em um programa-mql5 devem garantir a conformidade de chamadas do Windows API. Para garantir tal conformidade, no texto fonte de programas escritos em C ou C++, use a palavra-chave _stdcall, que é específico para os compiladores Microsoft(r). Esta conformidade é caraterizada pelo seguinte:

- Chamador (no nosso caso, é um programa-mql5) deve "ver" um proto-tipo de uma função chamada (importado da DLL), a fim de combinar parâmetros em uma pilha de forma apropriada.

© 2000-2024, MetaQuotes Ltd.

- Chamador (em nosso caso, é um programa-mql5) coloca parâmetros na pilha em uma ordem reversa, da direita para esquerda - nesta ordem uma função lê os parâmetros passados para ela.
- parâmetros são passados por valor, exceto aqueles explicitamente passados por referência (em nossos casos de strings)
- Uma função importada limpa a pilha de forma independente através da leitura dos parâmetros passados para ela.

Ao descrever o proto-tipo de uma função importada, parâmetros padrão podem ser usados.

Se a biblioteca correspondente é incapaz de ser carregada, ou existe uma proibição quanto ao uso da DLL, ou a função importada não é encontrada - o Expert Advisor interrompe sua operação com a mensagem apropriada "Expert Advisor stopped" no Diário (arquivo de log). Neste caso o Expert Advisor não rodará até ser reiniciado. Um Expert Advisor pode ser reiniciado como resultado de recompilação ou após a tabela de suas propriedades ser aberto e OK ser pressionado.

Passando Parâmetros

Todos os parâmetros de tipos simples são passados por valor a menos que sejam explicitamente indicados que eles devem ser passados por referência. Quando uma string é passada, o endereço do buffer da string copiada é passado; se uma string é passada por referência, o endereço do buffer desta string sem copiá-la é passado para a função importada da DLL.

Estruturas que contém arrays dinâmicos, strings, classes, outras estruturas complexas, bem como arrays estáticos ou dinâmicos dos objetos listados, não podem ser passados como parâmetro para uma função importada.

Ao passar um array para uma DLL, o endereço do começo do buffer de dados é sempre passado (independentemente da flag AS_SERIES). Uma função dentro de uma DLL não sabe nada sobre a flag AS_SERIES, o array passado é um array estático de um comprimento indefinido; um parâmetro adicional deve ser usado para especificar o tamanho do array.

© 2000-2024, MetaQuotes Ltd.

Erros em Tempo de Execução

O subsistema de execução do terminal cliente tem a oportunidade de salvar o código do erro em caso de ocorrência durante a execução de um programa MQL5. Existe um variável predefinida _LastError para cada programa executável MQL5.

Antes de iniciar a função OnInit, a variável _LastError é zerada. Em caso de uma situação de erro ocorrer durante a processamento de cálculos ou no processo de chamada de funções internas, o a variável _LastError recebe o código do erro correspondente. O valor armazenado nesta variável pode ser obtida usando a função GetLastError().

Existem diversos erros críticos cuja ocorrência faz com que um programa seja encerrado imediatamente:

- divisão por zero
- ir além dos limites de um array
- usando um ponteiro de objeto incorreto

© 2000-2024, MetaQuotes Ltd.

Testando Estratégias de Negociação

A ideia de negociação automática é atraente pelo fato de que o robô de negociação pode trabalhar sem parar 24 horas por dia, sete dias por semana. O robô não fica cansado, em dúvida, ou com medo, ele é totalmente livre de quaisquer problemas psicológicos. Basta formalizar de forma clara as regras de negociação e implementá-las nos algoritmos, e o robô está pronto para trabalhar incansavelmente. Mas primeiro, você deve se certificar de que as duas condição importantes seguintes estão satisfeitas:

- O Expert Advisor realiza operações de negociação em concordância com as regras do sistema de negociação;
- A estratégia de negociação, implementada no EA, demonstra lucro no histórico.

Para obter respostas para estas questões, nos voltamos para o Provador de Estratégia, incluído no terminal cliente MetaTrader 5.

Esta seção aborda as características do teste e otimização de programa no Provador de Estratégia:

- Function Limitations in the Strategy Tester
- Modos de Geração de Tick
- Simulação de Spread
- Usando ticks reais durante os testes
- As Variáveis Globais do Terminal Cliente
- O Cálculo de Indicadores durante um Teste
- Carregando Histórico durante Teste
- Teste Multi-Moeda
- Simulação de Tempo no Provador de Estratégia
- Objetos Gráficos no Teste
- A Função OnTimer() no Provador de Estratégia
- A Função Sleep() no Provador de Estratégia
- Usando o Provador de Estratégia para Problemas de Otimização em Cálculos Matemáticos
- A Sincronização de Barras no modo "Somente Preço de Abertura"
- A função IndicatorRelease() no Provador de Estratégia
- Tratamento de Evento no Provador de Estratégia
- Agentes de Teste
- A Troca de Dados entre o Terminal e o Agente
- Usando a Pasta Compartilhada para Todos os Terminais Cliente
- Usando DLLs

Limitações de memória e de espaço em disco na MQL5 Cloud Network

© 2000-2024, MetaQuotes Ltd.

Ao executar a otimização na MQL5 Cloud Network, há uma restrição: o Expert Advisor testado não pode gravar mais de 4 GB de informações no disco e usar mais de 4 GB de RAM. Se o limite for excedido, o agente de rede não conseguirá concluir o cálculo corretamente e você não receberá o resultado do teste. Além disso, você será cobrado pelo tempo já gasto nos cálculos.

Se você precisar obter informações de cada passagem da otimização, use enviar quadros sem gravar no disco. Para não usar operações de arquivo nos Expert Advisors durante os cálculos na MQL5 Cloud Network, você pode aplicar essa verificação:

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Restrições de trabalho das funções no testador de estratégias de negociação

Existem restrições no que diz respeito ao trabalho de algumas funções no testador de estratégias de negociação

Funções Comment(), Print() e PrintFormat()

A fim de aumentar o desempenho na otimização de parâmetros do expert, não são executadas as funções Comment(), Print() e PrintFormat(). A exceção é o uso dessas funções dentro do manipulador Oninit(). Isto torna mais fácil procurar as causas de erros quando eles ocorrem.

Funções Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Funções de interação com o "mundo exterior" Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() e WebRequest() não são executadas no testador de estratégias.

Modos de Geração de Tick

Um Expert Advisor é um programa, escrito em MQL5, que é rodado a cada instante em resposta a algum evento externo. O EA tem uma função correspondente (handler de evento) para cada evento pré-definido.

© 2000-2024, MetaQuotes Ltd.

O evento NewTick (mudança de preço) é o evento principal para o EA e, portanto, precisamos gerar um seqüência de ticks para testar o EA. Existem 3 modos de geração de tick implementada no Provador de Estratégia do terminal cliente MetaTrader 5:

- Cada Tick
- OHLC por minuto (preços OHLC, Open-High-Low-Close, com barras de minuto)
- Somente Preços de Abertura

O básico e mais detalhado é o modo "Cada Tick", os outros dois modos são simplificações do modo básico, e serão descritos em comparação com o modo "Cada Tick". Considere todos os três modos em ordem para entender as diferenças entre eles.

"Cada Tick"

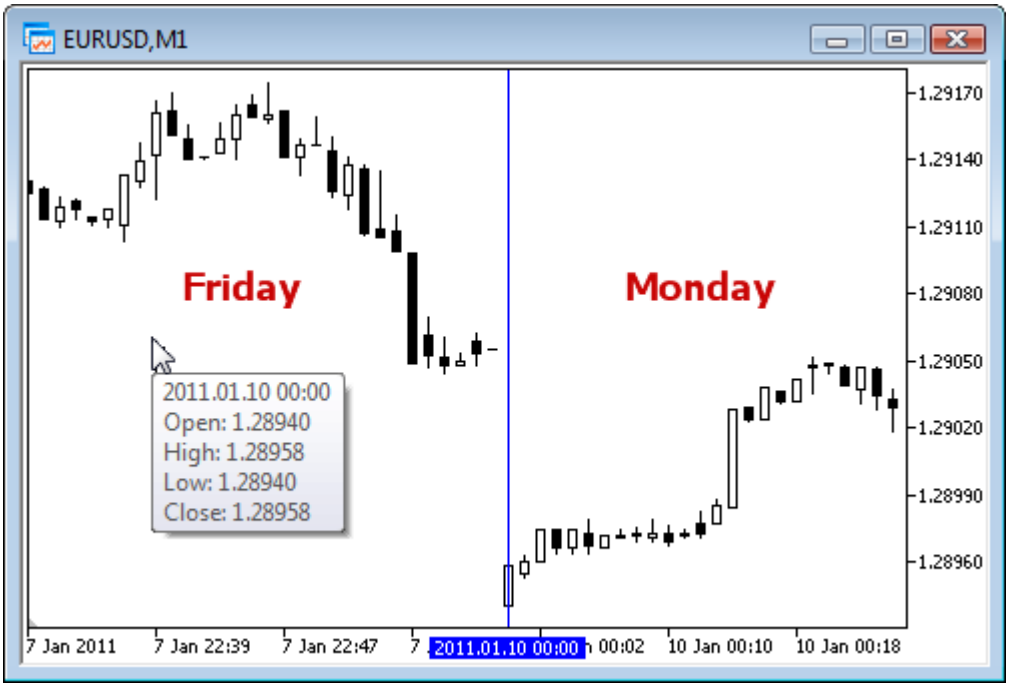
Os dados de cotação histórica para instrumentos financeiros são transferidos do servidor de negociação para o terminal cliente MetaTrader 5 na forma de barras de minuto empacotadas. Informações detalhadas sobre a ocorrência de solicitações e a construção das janelas de tempo requiridas podem ser obtidas a partir do capítulo Organizando Acesso a Dados da Referência MQL5.

O elemento mínimo do histórico de preços é a barra de minuto, a partir do qual pode-se obter informações sobre os quatro valores de preços:

- Abertura (Open) - o preço em que a barra de minuto foi aberta;
- Máximo (High) - o máximo que foi alcançado durante esta barra de minuto;
- Mínimo (Low) - o mínimo que foi alcançado durante esta barra de minuto;
- Fechamento (Close) - o preço de fechamento da barra.

A nova barra de minuto não é aberta no momento que o novo minuto começa (número de segundos se torna igual a 0), mas quando um tick ocorre - uma alteração de preço de no mínimo um ponto. A figura mostra a primeira barra de minuto da nova semana de negociação, que tem a hora de abertura de 2011.01.10 00:00. A diferença de preços entre Sexta e Segunda, que nós vemos no gráfico é comum, já que taxas de moeda flutuam mesmo em fins-de-semana em resposta ao surgimento de notícias.

© 2000-2024, MetaQuotes Ltd.



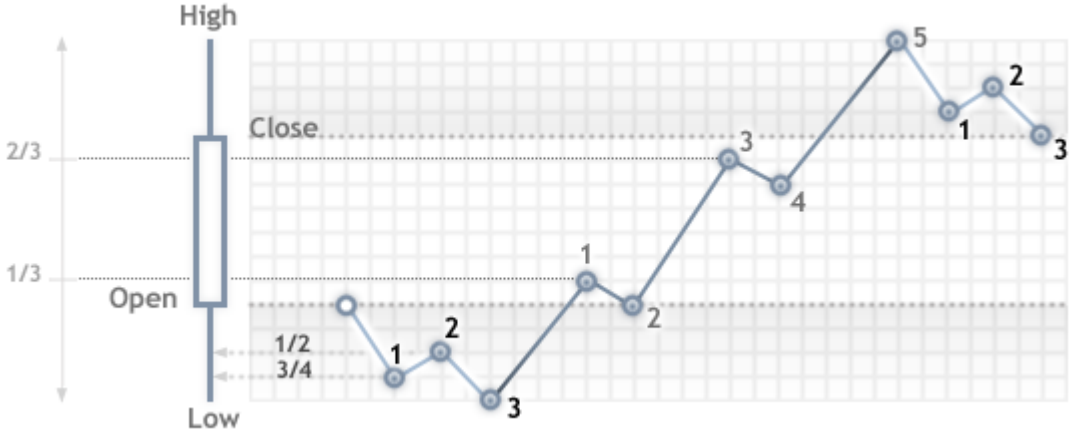
Para esta barra, nós sabemos apenas que a barra de minuto foi aberta em 10 de Janeiro de 2011 às 00 horas 00 minutos, mas nós não sabemos nada sobre os segundos. Ela poderia ter sido aberta às 00:00:12 ou 00:00:36 (12 ou 36 segundos após o inicio de um novo dia) ou qualquer outro instante dentro daquele minuto. Mas nós sabemos que o preço de Abertura do EURUSD foi de 1.28940 no instante de abertura da nova barra de minuto.

Nós também não sabemos, em termos de segundos, quando o tick, correspondente ao preço de fechamento da barra de minuto considerada, foi recebido. Nós só sabemos uma coisa - o último preço de Fechamento da barra de minuto. Para este minuto, o preço foi 1.28958. Os instantes de aparecimento dos preços de Máximo e Mínimo também são desconhecidos, mas nós sabemos que os preços máximo e mínimo foram nos níveis de 1.28958 e 1.28940, respectivamente.

Para testar a estratégia de negociação, nós precisamos de uma seqüência de ticks, na qual o trabalho do Expert Advisor será simulado. Assim, para toda barra de minuto, nós conhecemos os **4 pontos de controle**, onde os preços definitivamente estiveram. Se uma barra tem apenas 4 ticks, então isso é informação suficiente para realizar um teste, mas geralmente o volume de tick é maior que 4.

Portanto, existe a necessidade de gerar pontos de controle adicionais para ticks, que ocorreram entre os preços de Abertura, Máximo, Mínimo, e Fechamento. O princípio do modo de geração de ticks "Cada tick" é descrito dentro do O Algoritmo de Geração de Ticks dentro da Testador de Estratégia do Terminal MetaTrader 5, um número a partir do qual é apresentado a baixo.

© 2000-2024, MetaQuotes Ltd.



Ao testar no modo "Cada Tick", a função OnTick() do EA será chamado a cada ponto de controle. Cada ponto de controle é um tick de uma seqüência gerada. O EA receberá a hora e preço do tick simulado, assim como se estivesse trabalhando online.

> **Importante:** o modo de teste "Cada Tick" é o mais preciso, mas ao mesmo tempo, é o que consome mais tempo. Para um teste inicial da maioria das estratégias de negociação, é geralmente suficiente usar um dos outros dois modos de teste.

"OHLC por minuto"

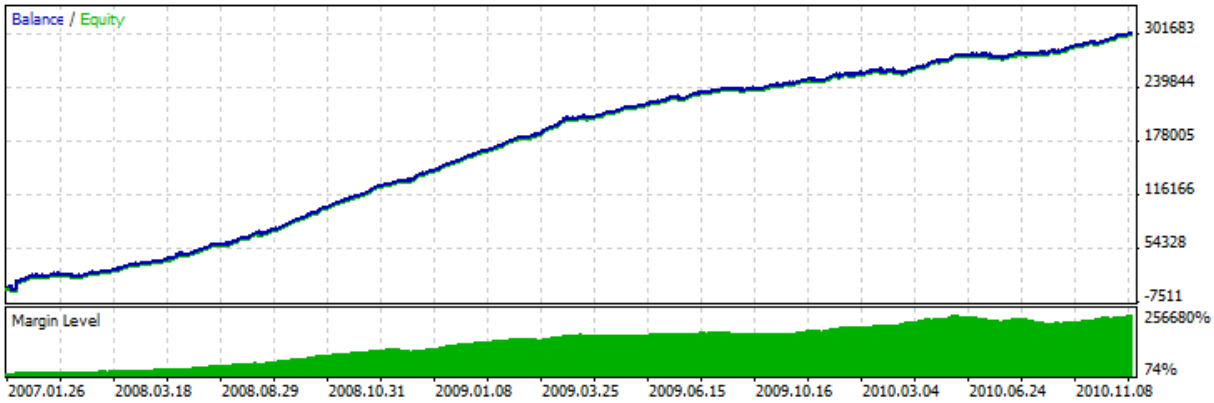
O modo "Cada Tick" é o mais preciso dos três modos, mas é ao mesmo tempo, o mais lento. A execução do handler OnTick() ocorre a cada tick, a mesmo tempo que o volume de tick pode ser muito grande. Para uma estratégia, na qual a seqüência de ticks da movimentação de preços em toda barra não importa, existe um modo de simulação mais rápido e grosseiro - "OHLC por minuto".

No modo "OHLC por minuto", a seqüência de ticks é construída somente pelos **preços OHLC das barras de minuto**, o número de pontos de controle gerados é significativamente reduzido - assim como o tempo de teste. O disparo da função OnTick() é realizada sob todos os pontos de controle, que são construídos pelos preços OHLC das barras de minuto.

A opção de não gerar ticks intermediários adicionais entre os preços de Abertura, Máximo, Mínimo e Fechamento, leva ao aparecimento de um rígido determinismo na evolução dos preços, a partir do momento que o preço de Abertura é determinado. Isso torna possível criar um "Teste Graal", que mostra um agradável gráfico ascendente do saldo do teste.

Um exemplo de tal Graal é apresentado no Código Base - Grr-al.

© 2000-2024, MetaQuotes Ltd.



A figura mostra um gráfico muito atrativo deste teste de EA. Como ele foi obtido? Nós conhecemos 4 preços para uma barra de minuto, e nós sabemos que o primeiro é preço de Abertura, e o último é o preço de Fechamento. Nós temos os preços Máximo e Mínimo entre eles, e a seqüência de suas ocorrências é desconhecida, mas é sabido que o preço Máximo é maior ou igual que o preço de Abertura (e o preço Mínimo é menor ou igual ao preço de Abertura).

É suficiente determinar o momento da recepção do preço de Abertura, e então analisar o próximo tick a fim de determinar que preço nós temos no momento - ou o Máximo ou o Mínimo. Se o preço for abaixo do preço de Abertura, então nós temos um preço Mínimo e comprarmos neste tick, o próximo tick corresponderá ao preço Máximo, no qual nós encerramos a compra e abrimos para venda. O próximo tick é o último, este é o preço de Fechamento, e encerramos a venda nele.

Se após o preço, nós recebermos um tick com um preço maior que o preço de Abertura, então a seqüência de operações (deals) é invertida. Processe uma barra de minuto neste modo "trapaceiro", e espere para pela próxima barra.

Ao testar tal EA no histórico, tudo vai bem, mas uma vez que lançamos ele online, a verdade começa a ser revelada - a linha de saldo permanece regular, porém em direção descendente. Para expor este truque, nós simplesmente precisamos executar o EA no modo "Cada Tick".

Observação: Se os resultados do teste do EA em modos de teste grosseiros ("OHLC por minuto" e "Somente Preços de Abertura") parecerem muito bons, certifique-se de testar-lo no modo "Cada Tick".

"Somente Preços de Abertura"

Neste modo os ticks são gerados baseados nos preços OHLC das janelas de tempo selecionados para teste. A função OnTick() do Expert Advisor roda somente no começo da barra, no preço de Abertura. Devido a este característica, níveis de stop e ordens pendentes podem disparar em um preço que difere do preço especificado (especialmente ao testar em janelas de tempo mais altas). Em contrapartida, nós temos uma oportunidade de executar rapidamente um teste de avaliação do Expert Advisor.

Os períodos W1 e MN1 são exceções no modo de geração de ticks "Somente Preços de Abertura": para estas janelas de tempo, ticks são gerados para os preços OHLC de cada dia, e não preços OHLC da semana ou mês.

Suponha que nós testemos um Expert Advisor sob EURUSD H1 no modo "Somente Preços de Abertura". Neste caso o número total de ticks (pontos de controle) não será mais que 4 * número de barras de

© 2000-2024, MetaQuotes Ltd.

uma-hora dentro do intervalo testado. Mas o handler **OnTick() é chamado somente na abertura da barra de uma-hora**. As verificações requeridas para um teste correto ocorrem no resto dos ticks (que são "escondidas" do EA).

- O cálculo de requerimentos de margem;
- O disparo de níveis de Stop Loss e Take Profit;
- O disparo de ordens pendentes;
- A remoção de ordens pendentes expiradas.

Se não existirem nenhum posição aberta ou ordens pendentes, nós não precisamos realizar destas verificações nos ticks escondidos, e o aumento da velocidade pode ser bastante substancial. Este modo "Somente Preços de Abertura" é bem adequado para testar estratégias que processam operações (deals) somente na abertura da barra e não usam ordens pendentes, bem como ordens Stop Loss e Take Profit. Para estratégias deste tipo, a necessidade de precisão do teste é preservada.

Vamos usar o Expert Advisor Moving Average (Média Móvel) do pacote padrão como exemplo de um EA, que pode ser testado em qualquer modo. A lógica deste EA é construído de tal forma que todas as decisões são feitas na abertura da barra, e operações (deals) são executadas imediatamente, sem o uso de ordens pendentes.

Executar um teste de EA no EURUSD H1 sobre um intervalo a partir de 2010.09.01 to 2010.12.31, e comparar os gráficos. A figura mostra o gráfico de saldo do relatório de teste para todos os três modos.

Como se pode ver, os gráficos nos diferentes modos de teste são exatamente o mesmo para o EA Moving Average do pacote padrão.

Existem algumas limitações no modo "Somente Preços de Abertura":

- Você não pode usar o modo de execução Atraso Aleatório.
- No Expert testado, você não pode acessar os dados do período inferior do período utilizado para o teste/otimização. Por exemplo, se você rodar um teste em M20, você não pode acessar dados de

© 2000-2024, MetaQuotes Ltd.

M30, mas é possível acessar H1. Além disso, os períodos superiores que são acessados devem ser múltiplos do período do teste. Por exemplo, se você rodar um teste em M20, você não pode acessar dados de M30, mas é possível acessar H1. Estas limitações estão relacionadas com a impossibilidade de se obter dados de períodos inferiores ou não múltiplos de fora das barras geradas durante o teste/otimização.

- Limitações no acesso a dados de outros períodos também se aplicam a outros ativos cujos dados são usados pelo Expert Advisor. Neste caso, a limitação para cada ativo depende do primeiro período acessado durante o teste/otimização. Suponha que durante teste em EURUSD H1, um Expert Advisor acesse dados de GBPUSD M20. Neste caso, o Expert será capaz de continuar a usar os dados de EURUSD H1, H2, etc, bem como GBPUSD M20, H1, H2, etc.

Observação: O modo "Somente Preços de Abertura" tem o mais rápido tempo de teste, mas não é adequado para todas as estratégias de negociação. Selecione o modo de teste desejado baseado nas características do sistema de negociação.

Para concluir a seção sobre os modos de geração de tick, vamos considerar uma comparação visual dos diferentes modos de geração de tick para EURUSD, para duas barras de M15 no intervalo de 2011.01.11 21:00:00 - 2011.01.11 21:30:00.

Os ticks foram salvos em arquivos diferentes usando o EA WriteTicksFromTester.mq5 e o fim desses nomes de arquivos são especificados nos parâmetros de entrada filenameEveryTick, filenameOHLC e filenameOpenPrice input-parameters.

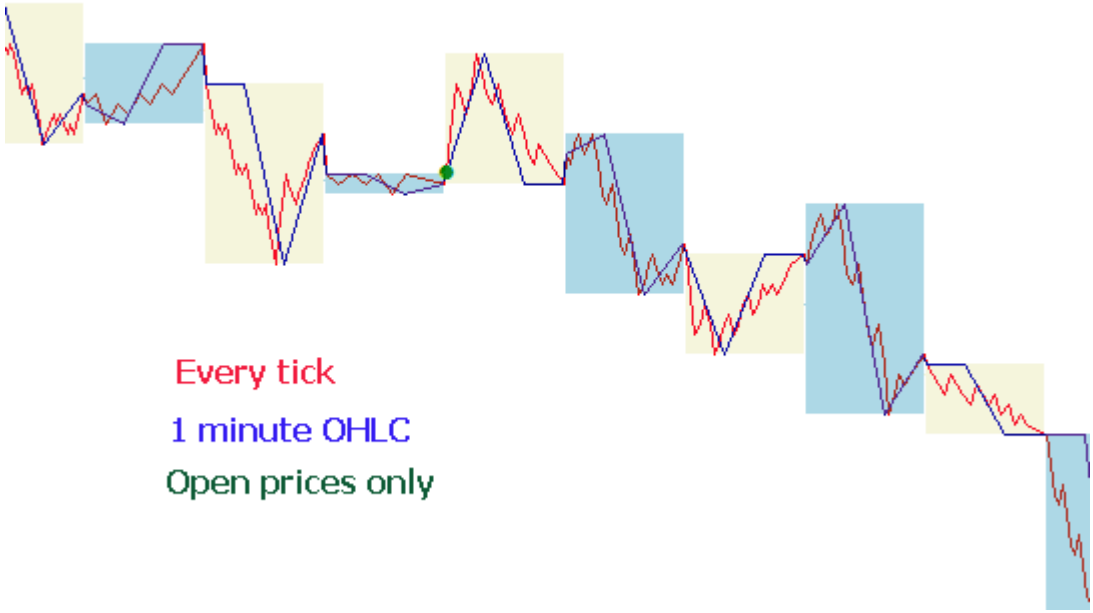
Strategy Tester

Variable	Value	Start	Step	Stop	Steps
start	2011.01.11 21:00:00	2011.01.11 21:00:00	1	2380.04.19 18:00:00	
end	2011.01.11 21:30:00	2011.01.11 21:30:00	1	2380.04.19 23:00:00	
filenameEveryTick	everytick.csv				
filenameOHLC	ohlc.csv				
filenameOpenPrice	openprice.csv				

Settings | Inputs | Optimization Results | Agents | Journal

Para obter três arquivos com três seqüências de tick (para cada um dos seguintes modos: "Cada Tick", "OHLC por minuto" e "Somente Preços de Abertura"), o EA foi disparado três vezes nos modos correspondentes, em execuções individuais. Então, os dados destes três arquivos foram exibidos no gráfico usando o indicador TicksFromTester.mq5. O código do indicador está anexado neste artigo.

© 2000-2024, MetaQuotes Ltd.



Por padrão, todos as operações de arquivo na linguagem MQL5 são feitas dentro da "caixa de areia de arquivo" e durante um teste o EA tem acesso somente a sua própria "caixa de areia de arquivo". A fim de que o indicador e o EA trabalhem com arquivos provenientes de uma pasta comum durante o teste, nós usados o flag FILE_COMMON. Um exemplo de código do EA:

```
//--- abre o arquivo
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- verifica a handle de arquivo
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Erro na abertura do arquivo %s para escrever. Erro de codigo=%d",fi
      return;
     }
   else
     {
      PrintFormat("O arquivo será criado na %s pasta",TerminalInfoString(TERMINAL_COM
     }
```

Para ler dos dados no indicador, nós também usamos o flag FILE_COMMON. Isso nós permite evitar transferir manualmente os arquivos necessários de uma pasta para outra.

```
//--- abre o arquivo
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- verifica a handle de arquivo
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Erro na abertura do arquivo %s para leitura. Erro de codigo=%d",fna
      return;
     }
   else
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("Arquivo será aberto a partir %s",TerminalInfoString(TERMINAL_COMMON
    }
```

Simulação de Spread

A diferença de preço entre os preços de Venda (Bid) e Compra (Ask) é chamado de spread. Durante um teste, o spread não é modelado mas é obtido a partir de dados históricos. If the spread is less than or equal to zero in the historical data, then the last known (at the moment of generation) spread of is used by testing agent.

No Provador de Estratégia, o spread é sempre considerado flutuante. Isto é, a função SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) sempre retorna true.

Além disso, os dados históricos contém valores de tick e volumes de negociação. Para o armazenamento e recuperação dos dados nós usamos a estrutura especial MqlRates:

```
struct MqlRates
  {
   datetime time;         // Hora de início do período
   double   open;         // Preço de Abertura
   double   high;         // O mais alto preço do período
   double   low;          // O mais baixo preço do período
   double   close;        // Preço de Fechamento
   long     tick_volume;  // Volume de Tick
   int      spread;       // Spread
   long     real_volume;  // Volume de Negociação
  };
```

Usando ticks reais durante os testes

O teste e a otimização de acordo com ticks reais se aproximam ao máximo das condições reais. Em vez de ticks gerados com base em dados de minuto, são usados ticks reais acumulados pela corretora. Esses são ticks provindos da bolsa e dos provedores de liquidez.

Para garantir a maior precisão - durante os testes em modo de ticks reais - também são usadas as barras de minuto. Nelas são verificados e corrigidos os dados de ticks. Isto também evita a divergência de gráficos no testador e no terminal de cliente.

O testador verifica a correspondência entre os dados de tick e os parâmetros da barra de minuto, isto é: o tick não deve ultrapassar os preços High/Low da barra; o tick que a abre e fecha o minuto deve coincidir com os preços Open/Close da barra. Também é comparado o volume. Se identificada uma diferença, são descartados todos os ticks correspondentes a esta barra de minuto. Em vez deles, serão utilizados os ticks gerados (como no modo "Cada tick").

Se no histórico do símbolo existir uma barra de minuto, mas, se, nesse minuto, não houver dados de ticks, o testador gerará ticks no modo "Todos os ticks". Isto permite plotar corretamente o gráfico no testador, caso a corretora tenha os dados de tick incompletos.

Se, no histórico do símbolo, não existir uma barra de minuto, mas, se, nesse minuto, houver dados de ticks, então, esses ticks podem ser utilizados. Por exemplo, as barras de símbolos de bolsa são formados de acordo com os preços Last. Se, a partir do servidor, chegarem apenas ticks com preços

© 2000-2024, MetaQuotes Ltd.

Bid/Ask sem preço Last, a barra não será formada. O testador usará estes dados de ticks, uma vez que eles não contradizem as de minuto.

Os dados de ticks podem ser diferentes das barras de minuto por várias razões. Por exemplo, devido a desconexões ou outras falhas, quando dados são transferidos da fonte para o terminal de cliente. Ao testar, os dados de minuto são considerados mais confiáveis.

Ao testar ticks reais, considere as seguintes particularidades:

- Ao executar o teste, não só são sincronizados os dados de ticks, mas também os de minuto de acordo com o instrumento.
- Os ticks são armazenados no cache do símbolo no testador de estratégias. O tamanho do cache é inferior a 128 000 ticks. Após chegarem novos ticks, seus dados mais antigos serão empurrados para fora. No entanto, com ajuda da função CopyTicks podem ser obtidos ticks fora do cache (somente nos testes em ticks reais). Neste caso, os dados serão solicitados a partir da base de ticks do testador, ela corresponde completamente à base de dados do terminal de cliente. Nesta base não é feito nenhum ajuste de barras de minuto. Portanto, os ticks nele podem ser diferentes dos ticks no cache.

As Variáveis Globais do Terminal Cliente

Durante um teste, as variáveis globais do terminal cliente são também emulados, mas elas não estão relacionadas como as variáveis globais correntes do terminal, que podem ser vistas no terminal usando a tecla F3. Significa que todas as operações com as variáveis globais do terminal, durante um teste, acontecem fora terminal cliente (no agente de teste).

O Cálculo de Indicadores durante um Teste

No modo tempo-real, os valores dos indicadores são calculados a cada tick.

No testador de estratégia, os indicadores são calculados apenas quando são acessados à procura de dados, ou seja, apenas no momento em que os valores dos buffers dos indicadores são solicitados. A exceção a isto é quando indicadores personalizados são definidos como #property tester_everytick_calculate, neste caso, o recálculo é feito a cada tick.

No modo de teste visual, todos os indicadores são recalculados incondicionalmente com a chegada de um novo tick, de modo a aparecerem corretamente no gráfico de teste visual.

O cálculo do indicador a cada tick é feito uma vez e todas as subsequentes solicitações de dados do indicador antes da chegada de um novo tick não provocarão um recálculo. Por isso, se, no Expert Advisor, o timer estiver ativo com a função EventSetTimer(), antes da chamada do handler OnTimer() serão solicitados os dados do indicador desde a recepção do último tick. Se o indicador ainda não tiver sido calculado no último tick, será iniciado o cálculo dos valores do indicador. Se os dados já tiverem sido preparados, eles serão fornecidos sem um novo recálculo.

Assim, todos os cálculos do indicador são feitos da forma mais econômica possível, quer dizer, se o indicador já tiver sido calculado em um determinado tick, os dados do indicador serão fornecidos como estão, ou seja, o indicador não é será recalculado.

Carregando Histórico durante Teste

© 2000-2024, MetaQuotes Ltd.

O histórico de um ativo a ser testado é sincronizado e carregado pelo terminal a partir do servidor de negociação antes de começar o processo de teste. Durante a primeira vez, o terminal carrega todo o histórico disponível de um ativo de forma a não requerer este histórico mais tarde. Após isso, somente dados novos são carregados.

Um agente de teste recebe o histórico de um ativo a ser testado a partir do terminal cliente logo após o começo do teste. Se os dados de outros instrumentos são usados no processo de teste (por exemplo, é um Expert Advisor multi-moeda), o agente de teste solicita o histórico requerido do terminal cliente durante a primeira chamada a tais dados. Se os dados históricos estiverem disponíveis no terminal, eles são imediatamente passados para o agente de teste. Se os dados não estiverem disponíveis, o terminal solicita e os baixa do servidor, e então os passa para o agente de teste.

Dados de instrumentos adicionais também são requeridos para calcular taxas-cruzadas de operações de negociação. Por exemplo, ao testar uma estratégia em EURCHF com a moeda de depósito em USD, antes de processar a primeira operação de negociação, o agente de teste solicita os dados históricos de EURUSD e USDCHF do terminal cliente, embora a estratégia não contenha chamadas de uso direta a estes ativos.

Antes de testar um estratégia multi-moeda, é recomendável baixar todos os dados históricos necessários para o terminal cliente. Isso ajudará a evitar atrasos em um teste/otimização associado com a baixa dos dados requeridos. Você pode baixar histórico, por exemplo, via abertura dos gráficos apropriados e rolá-los para o começa do histórico. Um exemplo de carga forçada de histórico para o terminal está disponível na seção Organizando Acesso a Dados da Referência MQL5.

Agentes de teste, por sua vez, recebem o histórico do terminal no formato compactado. Durante o próximo teste, o Provador de Estratégia não carrega o histórico do terminal, porque os dados requeridos estão disponíveis desde a execução anterior do Provador de Estratégia.

- O termianl carrega o histórico a partir de um servidor de negociação apenas uma vez, na primeira vez que o agente solicita o histórico de um ativo a ser testado do terminal. O histórico é carregado em forma de pacote para reduzir o tráfego.
- Ticks não são enviados pela rede, eles são gerados nos agentes de teste.

Teste Multi-Moeda

O Provador de Estratégia nos permite realizar um teste de estratégias, negociando em múltiplos ativos. Tais EAs são convencionalmente referidos como Expert Advisors multi-moedas, já que originalmente, um teste era realizado somente para um único ativo. No Provador de Estratégia do terminal MetaTrader 5, nós podemos modelar negociação para todos os ativos disponíveis.

O Provador de Estratégia carrega o histórico dos ativos usados do **terminal cliente** (não do servidor de negociação!) automaticamente durante a primeira chamada aos dados do ativo.

O agente de teste baixa somente o histórico faltante, com uma pequena margem para fornecer os dados necessários no histórico para o cálculo dos indicadores no período inicial do teste. Para as janelas de tempo D1 ou menor, o volume mínimo de histórico baixado é de um ano.

Assim, se nós rodarmos um teste no intervalo 2010.11.01-2010.12.01 (teste para um intervalo de um mês) com um período de M15 (cada barra e igual a 15 minutos), então será solicitado ao terminal o histórico do instrumento para o ano inteiro de 2010. Para a janela de tempo semanal, nós solicitaremos um histórico de 100 barras, que é aproximadamente dois anos (um ano tem 52

© 2000-2024, MetaQuotes Ltd.

semanas). Para testar em uma janela de tempo mensal, o agente solicitará o histórico de 8 anos (12 meses x 8 anos = 96 meses).

Se não houver barras necessárias, a **data de ínicio do teste será automaticamente deslocada** do passado para o presente para fornecer a reserva necessária de barras antes do teste.

Durante um teste, a janela "Observação de Mercado" é também emulada, a partir do qual se pode obter informações sobre os ativos.

Por padrão, no começo de um teste, existe somente um ativo na "Observação de Mercado" do Provador de Estratégia - o ativo em que o teste está rodando. Todos os ativos necessários são conectados à janela "Observação de Mercado" do Provador de Estratégia (não ao terminal!) automaticamente quando referenciados.

> Antes de começar um teste de um Expert Advisor multi-moeda, é necessário selecionar os ativos requeridos para o teste na janela "Observação de Mercado" do terminal e carregar os dados requeridos. Durante a primeira chamada de um ativo "estrangeiro", seu histórico será automaticamente sincronizado entre o agente de teste e o terminal cliente. Um ativo "estrangeiro" é o outro ativo, diferente daquele em que o teste é rodado.

Referência aos dados de um "outro" ativo ocorre nos seguintes casos:

- Ao usar a função de indicadores técnicos e IndicatorCreate() no ativo/período;
- A solicitação aos dados da janela "Observação de Mercado" por de um outro ativo:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- Solicitação da série de tempo para um ativo/janela de tempo atráves do uso das seguintes funções:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose

© 2000-2024, MetaQuotes Ltd.

8. CopyTickVolume
9. CopyRealVolume
10. CopySpread

No momento da primeira chamada a um "outro" ativo, o processo de teste é interrompido e o histórico é baixado para o ativo/período, do terminal para o agente de teste. Ao mesmo tempo, a geração da seqüência de tick para este ativo é feita.

Uma seqüência de tick individual é gerada para cada ativo, de acordo com o modo de geração de tick selecionado. Você também pode solicitar explicitamente o histórico dos ativos desejados através de chamada ao SymbolSelect() no handler OnInit() - a baixa do histórico será feita imediatamente antes do teste do Expert Advisor.

Assim, não é necessário nenhum esforço extra para realizar teste de multi-moeda no terminal cliente MetaTrader 5. Apenas abra os gráficos dos apropriados ativos no terminal cliente. O histórico será automaticamente carregado do servidor de negociação para todos os ativos requeridos, contanto que ele contenha estes dados.

Simulação de Tempo no Provador de Estratégia

Durante um teste, a hora local TimeLocal() é sempre igual a hora do servidor TimeTradeServer(). Por sua vez, a hora do servidor é sempre igual a hora correspondente à hora GMT - TimeGMT(). Desta forma, todas estas funções mostram a mesma hora durante um teste.

A ausência de diferença entre as horas GMT, local, e do servidor no Provador de Estratégia é feita deliberadamente para o caso de não haver conexão com o servidor. Os resultados do teste devem sempre ser os mesmos, independentemente de haver ou não uma conexão. Informações sobre a hora do servidor não é armazenado localmente, e é obtido do servidor.

Objetos Gráficos no Teste

Durante um teste/otimização objetos gráficos não são plotados. Assim, ao referenciar as propriedades de um objeto criado durante um teste/otimização, um Expert Advisor receberá valores zero.

> Este limitação não se aplica ao se testar em modo visual.

A Função OnTimer() no Provador de Estratégia

MQL5 fornece a oportunidade de tratar eventos de timer. A chamada ao handler OnTimer() é feita independentemente do modo do teste. Isso significa que se um teste é rodado no modo "Somente Preços de Abertura" para o período H4, e o EA tem um timer definido para uma chamada por segundo, então na abertura de cada barra de H4, o handler OnTick() será chamado uma vez, e o handler OnTimer() será chamado 14400 vezes (3600 segundos * 4 horas). A quantidade de tempo de teste do EA que será aumentada depende da lógica do EA.

Para verificar a dependência do tempo de teste a partir de um dada freqüência do timer, nós temos que criar um EA simples sem nenhum operação de negociação.

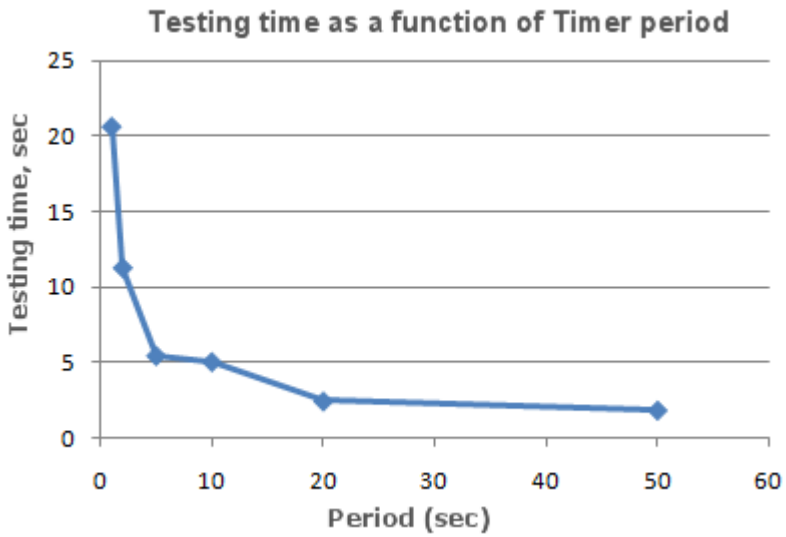
```
//--- parâmetros de entrada
```

© 2000-2024, MetaQuotes Ltd.

```
input int      timer=1;              // valor do timer, segundos
input bool     timer_switch_on=true; // timer ativado
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- rodar o timer se timer_switch_on==true
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- para o timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Função Timer                                                     |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// nenhum ação é tomada, o corpo do handler está vazio
  }
//+------------------------------------------------------------------+
```

As medições de tempo de tempo são obtidas em valores diferentes do parâmetro timer (periodicidade do evento Timer). Sob os dados obtidos, nós plotamos o tempo do teste como função do período do Timer.

© 2000-2024, MetaQuotes Ltd.



Pode-se ver claramente que quanto menor é o parâmetro do timer, durante a a inicialização da função EventSetTimer, menor é o período entre as chamadas do handler OnTimer(), e maior é o tempo de teste, sob as mesmas outras condições.

A Função Sleep() no Provador de Estratégia

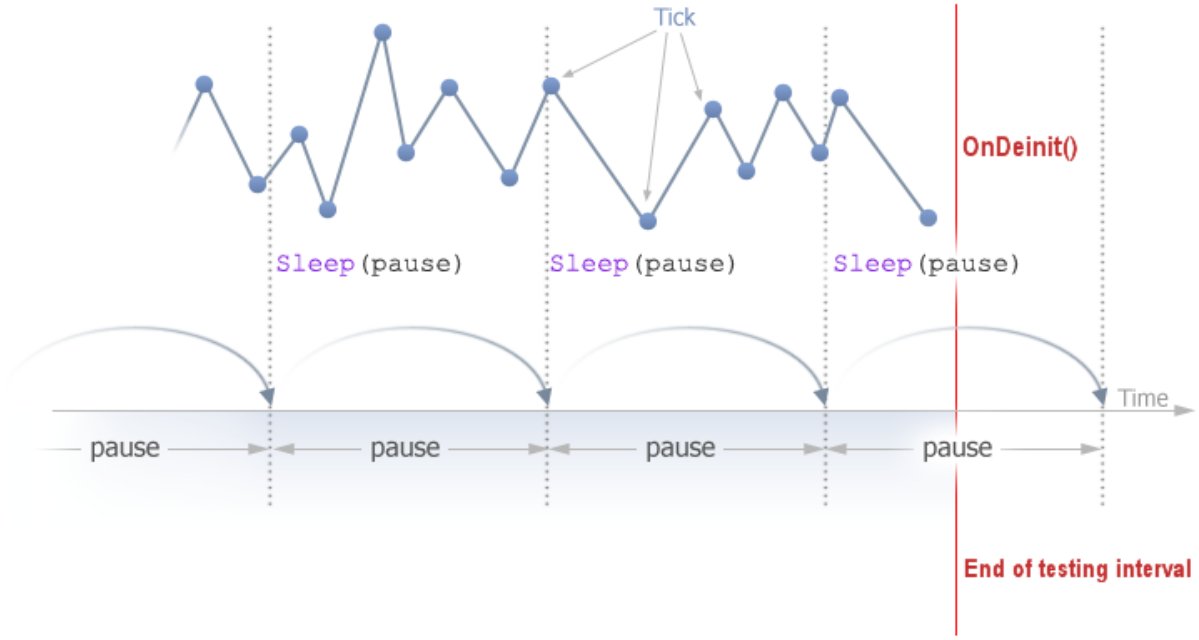
A função Sleep() permite ao EA ou script suspender a execução do programa MQL5 por um tempo, ao trabalhar no gráfico. Isso pode ser útil ao solicitar dados, que não estão prontos no momento da solicitação e você precisa esperar até eles estarem prontos. Um exemplo detalhado do uso da função Sleep() pode ser encontrado na seção Preparativo para acesso a dados.

O processo de teste não é atrasado por chamadas à função Sleep(). Quando você chama o Sleep(), os ticks gerados são "jogados" com um atraso especificado, que pode resultar no disparo de ordens pendentes, stops, etc. Após uma chamada do Sleep(), o tempo simulado no Provador de Estratégia aumenta em um intervalo, especificado no parâmetro da função Sleep.

Se como resultado da execução da função Sleep(), o tempo corrente no Provador de Estratégia passar do período de teste, então você receberá o erro "Infinite Sleep loop detected while testing". Se você receber este erro, os resultados do teste não serão rejeitados, todas as computações são realizadas em seus volumes completos (o número de operações (deals), abrandamento, etc) e os resultados deste teste são transmitidos para o terminal.

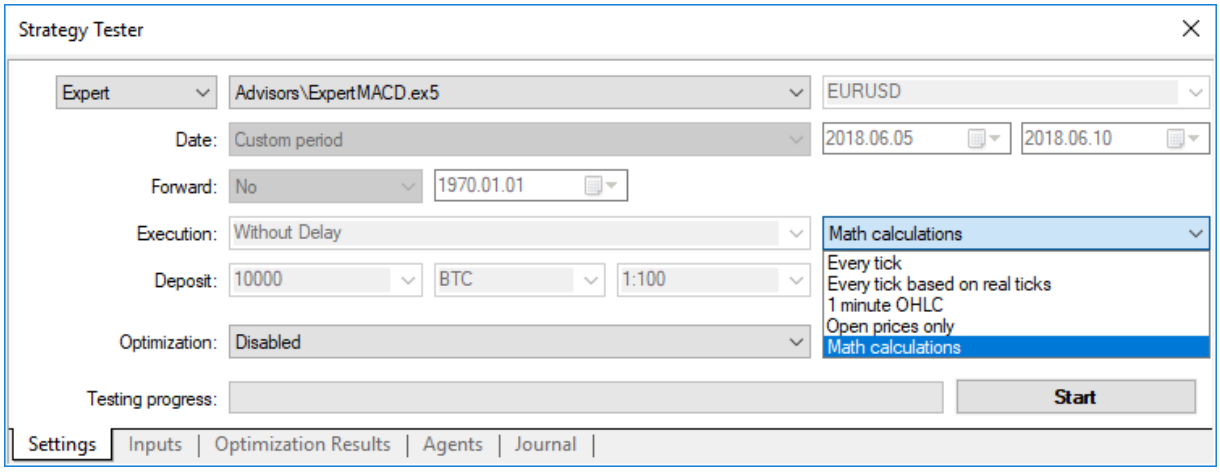
A função Sleep() não funcionará em OnDeinit(), já que após ele ser chamado, será certo que o tempo de teste ultrapassará a faixa do intervalo de teste.

© 2000-2024, MetaQuotes Ltd.



Usando o Provador de Estratégia para Problemas de Otimização em Cálculos Matemáticos

O Provador de Estratégia no terminal MetaTrader 5 pode ser usado, não apenas para testar estratégias de negociação, mas também para cálculos matemáticos. Para usar-lo, é necessário selecionar o modo "Cálculos matemáticos"



Neste caso, somente três funções serão chamadas: OnInit(), OnTester(), OnDeinit(). No modo "Cálculos matemáticos" o Provador de Estratégia não gera nenhum tick e não baixa o histórico.

O Provador de Estratégia também trabalha em modo "Cálculos matemáticos" se você especificar a data inicial maior que a data final.

Ao usar o Provador de Estratégia para resolver problemas matemáticos, a carga do histórico e a geração de ticks não ocorrem.

Um problema matemático típico para ser revolvido no Provador de Estratégias do MetaTrader 5 - busca de um extremo de uma função com muitas variáveis.

© 2000-2024, MetaQuotes Ltd.

Para resolvê-lo nos precisamos que:

- O cálculo do valor da função deve estar localizado na função OnTester();
- Os parâmetros da função devem estar definidos como variáveis de entrada do Expert Advisor;

Compilar o EA, abrir a janela "Provador de Estratégia". Na guia "Parâmetros de entrada", selecionar as requeridas variáveis de entrada, e definir o conjunto de valores de parâmetros por meio da especificação dos valores iniciar, parar, e passo de cada uma das variáveis da função.

Selecionar o tipo de otimização - "Algoritmo completo lento" (busca completa do espaço de parâmetros) ou "Rápido (algoritmo genético)". Para uma busca simples do extremo da função, é melhor escolher uma otimização rápida, mas se você quiser calcular os valores para conjunto inteiro de variáveis, então é melhor usar a otimização lenta.

Selecione o modo "Cálculo matemático" e usando o botão "Iniciar", executar a procedimento de otimização. Note que ao otimizar, o Provador de Estratégia buscará pelos valores máximos da função OnTester. Para encontrar um mínimo local, retorne o inverso do valor computado da função proveniente da função OnTester:

```
return(1/function_value);
```

É necessário verificar que o function_value não é igual a zero, já que de outra forma nós podemos obter um erro crítico de dividir por zero.

Existe uma outra forma, mais conveniente e que não distorce os resultados da otimização, ela foi sugerida pelos leitores deste artigo:

```
return(-function_value);
```

Esta opção não requer a verificação do function_value ser igual a zero, e a superfície dos resultados da otimização em uma representação em 3D tem a mesma forma, porém é espelhada da original.

Como exemplo, fornecemos a função sink():

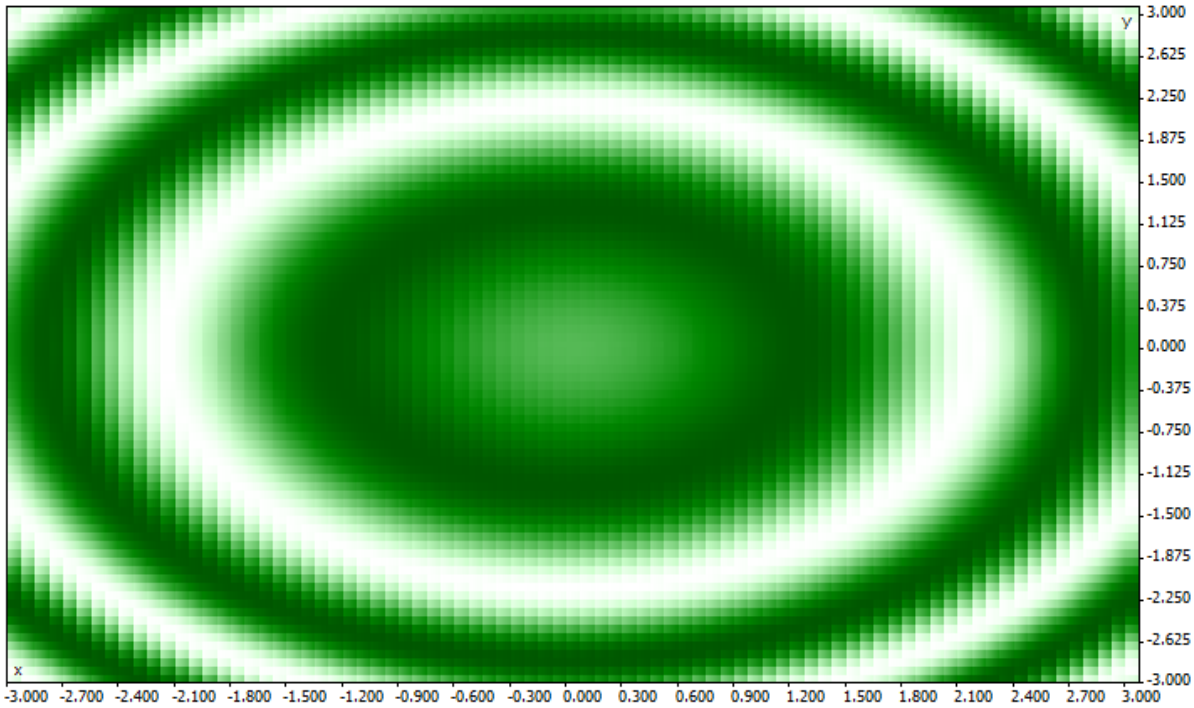
$$sink(x,y) = \sin\left(x^2 + y^2\right)$$

O código do EA para encontrar o extremo desta função é colocada dentro do OnTester():

```
//+------------------------------------------------------------------+
//|                                                        Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parâmetros de entrada
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
```

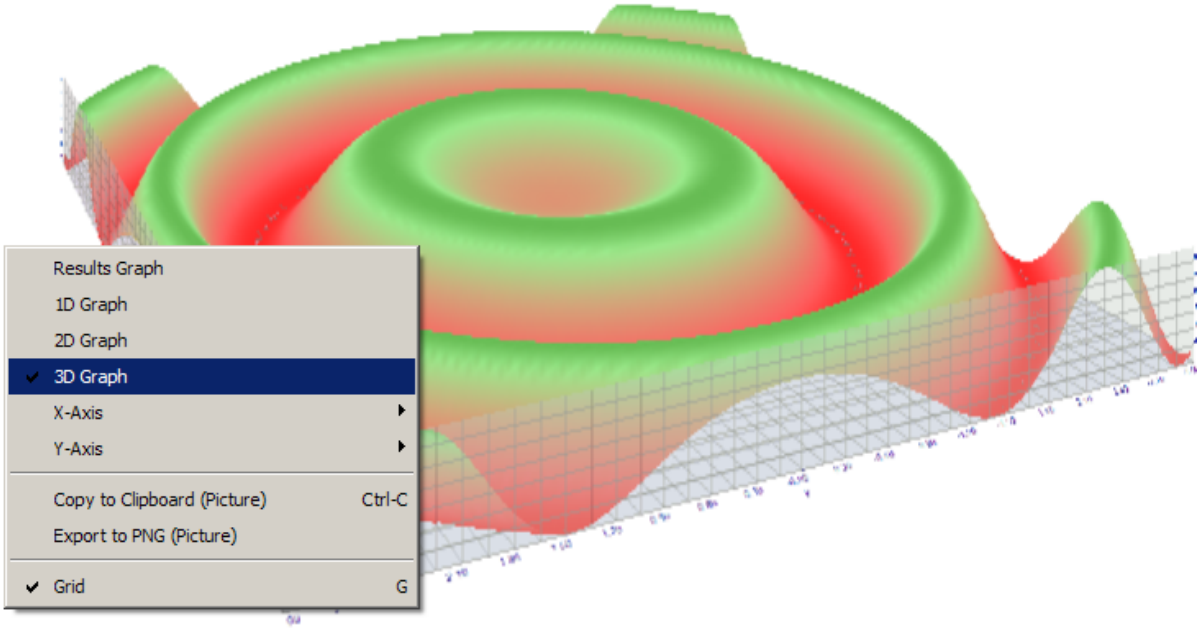
```
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

Execute uma otimização e veja os resultados da otimização na forma de um gráfico 2D.



Quanto melhor for o valor para um dado par de parâmetros (x,y), mais saturada é a cor. Como era esperado a partir da observação da forma da formula sink(), seus valores forma círculos concêntricos com centro em (0,0). Pode-se ver no gráfico 3D, que a função sink() não tem um único extremo global:

© 2000-2024, MetaQuotes Ltd.



A Sincronização de Barras no modo "Somente Preço de Abertura"

O Provador de Estratégia no terminal cliente do Metatrader 5 permite-nos verificar os chamados EAs "multi-moedas". Uma EA multi-moeda - é um EA que negocia sobre dois ou mais ativos.

O teste de estratégias, que negociam em ativos múltiplos, impõe alguns requerimentos técnicos adicionais ao Provador de Estratégia:

- A geração de ticks para estes ativos;
- O cálculo de valores de indicadores para estes ativos;
- O cálculo de requerimentos de margem para estes ativos;
- Sincronização das seqüências de ticks geradas para todos os ativos negociados.

O Provador de Estratégia gera e joga uma seqüência de ticks para cada instrumento em concordância com o modo de negociação selecionado. Ao mesmo tempo e para cada ativo, uma nova barra é aberta, independentemente de como a barra abriu em outro ativo. Isso significa que ao testar um EA multi-moeda, uma situação pode ocorrer (e freqüentemente acontece), quando para um instrumento, uma nova barra já abriu, e para o outro instrumento, ela não abriu. Assim, em um teste, tudo acontece como na realizada.

Esta simulação autêntica do histórico no Provador de Estratégia não causa nenhum problema contanto que os modos de teste "Cada Tick" e "OHLC por minuto" sejam usados. Para estes modos, ticks suficientes são gerados para um candlestick, para ser capaz de esperar até a sincronização das barras de diferentes ativos aconteça. Mas como nós testamos estratégias multi-moeda no modo "Somente Preços de Abertura", se a sincronização das barras nos instrumentos de negociação é obrigatória? Neste modo, o EA é chamado somente em um tick, que corresponde ao momento de abertura das barras.

Nós ilustraremos isso com um exemplo: se nós estamos testando um EA no EURUSD, e um novo candlestick de hora abriu em EURUSD, então podemos reconhecer facilmente este fato - em um teste no modo "Somente Preços de Abertura", o evento NewTick corresponde ao momento da abertura da

© 2000-2024, MetaQuotes Ltd.

barra sobre o período do teste. Mas não exitem garantias que o novo candlestick abriu no ativo USPJPY, é que usado no EA.

Em condições normais, é suficiente completar o trabalho da função OnTick() e verificar pelo surgimento de uma nova barra em USDJPY no próximo tick. Mas ao testar em modo "Somente Preços de Abertura", não haverá um outro tick, e assim pode parecer que este modo não é adequado para testar EAs multi-moedas. Mas isso não é o caso - não se esqueça que o testador no MetraTrader 5 se comporta como se fosse no mundo real. Você pode esperar até uma nova barra ser aberta em outros ativos usando a função Sleep()!

O código do EA Synchronize_Bars_Use_Sleep.mq5, que mostra um exemplo da sincronização de barras no modo "Somente Preços de Abertura":

```
//+------------------------------------------------------------------+
//|                                   Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parâmetros de entrada
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verifica ativo
   if(_Symbol==other_symbol)
     {
      PrintFormat("Você tem de especificar outro símbolo nos parâmetros de entrada ou
      //--- força parar teste
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- variável estática usada para armazenar a hora da última barra
   static datetime last_bar_time=0;
//--- flag de sincronização
   static bool synchonized=false;
//--- caso a variável estática não esteja inicializada
   if(last_bar_time==0)
     {
```

```
      //--- é a primeira chamada, salva hora da barra e saiu
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("A variável last_bar_time é inicializada com valor %s",TimeToString
     }
//--- obtém hora de abertura da última barra do ativo do gráfico
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- se as horas não são iguais
   if(curr_time!=last_bar_time)
     {
      //--- salva a hora da barra aberta na variável estática
      last_bar_time=curr_time;
      //--- não sincronizado
      synchonized=false;
      //--- imprime mensagem
      PrintFormat("Uma nova barra está aparecendo no símbolo %s em %s",_Symbol,TimeToS
     }
//--- hora de abertura da barra de outra ativo
   datetime other_time;
//--- loop até a hora de abertura do outro ativo ser igual a curr_time
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("Espera 5 segundos..");
      //--- espera 5 segundos e chama SeriesInfoInteger(other_symbol,Period(),SERIES_L
      Sleep(5000);
     }
//--- barras são sincronizadas
   synchonized=true;
   PrintFormat("Tempo de abertura da barra do símbolo %s do gráfico: é %s",_Symbol,Tim
   PrintFormat("Tempo de abertura da barra do símbolo %s: é %s",other_symbol,TimeToStr
//--- TimeCurrent() não é útil, usar TimeTradeServer()
   Print("As barras são sincronizadas em ",TimeToString(TimeTradeServer(),TIME_SECONDS
  }
//+------------------------------------------------------------------+
```

Perceba que a última linha no EA, que exibe a hora corrente quando o sincronização de fato foi estabelecida:

```
   Print("As barras são sincronizadas em ",TimeToString(TimeTradeServer(),TIME_SECONDS
```

Para exibir a hora corrente usamos a função TimeTradeServer() ao invés de TimeCurrent(). A função TimeCurrent() retorna o hora do último tick, que não se altera após usar Sleep(). Executa o EA no modo "Abertura de preços apenas",e você verá uma mensagem sobre a sincronização dos barras.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

© 2000-2024, MetaQuotes Ltd.

Use a função TimeTradeServer() ao invés de TimeCurrent(), se você precisar obter hora corrente do servidor, e não a hora da chegado do último tick.

Existe uma outra forma para sincronizar barras - usando um timer. Um exemplo de tal EA é Synchronize_Bars_Use_OnTimer.mq5, que está anexado neste artigo.

A função IndicatorRelease() no Provador de Estratégia

Após completar um teste individual, um gráfico do instrumento é automaticamente aberto, que exibe as operações (deals) completas e os indicadores usados no EA. Isso ajuda a verificar visualmente os pontos de entrada e saída, e comprará-los com os valores dos indicadores.

> **Observação:** indicadores, exibidos no gráfico, que automaticamente abrem após a conclusão do teste, são calculados novamente após a conclusão do teste. Mesmo se estes indicadores foram usados no EA testado.

Mas em alguns casos, o programador pode querer ocultar a informação sobre quais indicadores estavam envolvidos nos algoritmos de negociação. Por exemplo, o código do EA é alugado ou vendido como um arquivo executável, sem o fornecimento do código fonte. Para este propósito, a função IndicatorRelease() é apropriada.

Se o terminal define um template com o nome tester.tpl no directory/profiles/templates do terminal cliente, então ele será aplicado ao gráfico aberto. Na usa ausência, o template padrão é aplicado. (default.tpl).

A função IndicatorRelease() é originalmente destinada para liberar a porção de cálculo do indicador, caso ele mais mais seja necessário. Isso permite economizar tanto a memória quando os recursos de CPU, porque cada tick pede um cálculo do indicador. Seu segundo propósito - é proibir a exibição de um indicador no gráfico de teste, após uma execução de teste individual.

Para proibir a exibição do indicador no gráfico após um teste, chame o IndicatorRelease() com o handle do indicator no handler OnDeinit(). A função OnDeinit() é sempre chamada após a conclusão e antes da exibição do gráfico de teste.

```
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() completado com sucesso");
   else Print("IndicatorRelease() retorna false. Erro de código ",GetLastError());
  }
```

A fim de proibir a exibição do indicador no gráfico, após a conclusão de um teste único, use a função IndicatorRelease() no handler OnDeinit().

Tratamento de Evento no Provador de Estratégia

© 2000-2024, MetaQuotes Ltd.

A presença do handler OnTick() no EA não é obrigatória para que possa ser passível de teste em dados históricos no Provador de Estratégias do MetaTrader 5. É suficiente para o EA conter pelo menos as seguintes funções-handlers:

- OnTick() - Handler de evento de uma nova chegada de tick;
- OnTrade() - Handler de evento de negociação;
- OnTimer() - Handler de evento de uma chegada de sinal do timer;
- OnChartEvent() - um handler para eventos de cliente.

Ao testar um EA, nós pode tratar eventos customizados usando a função OnChartEvent(), mas nos indicadores, esta função não pode ser chamada no testador. Mesmo se o indicador tiver o handler de evento OnChartEvent() e este indicador for usado em um EA testado, o indicador em si não receberá nenhum evento customizado.

Durante um teste, um Indicador pode gerar eventos customizados usando a função EventChartCustom(), e o EA pode processar este evento no OnChartEvent().

Além destes eventos, eventos especiais associados com o processo de teste e otimização são gerados no Provador de Estratégia:

- Tester - este evento é gerado após conclusão do teste do Expert Advisor sobre dados históricos. O evento Tester é tratado usando a função OnTester(). Esta função pode ser usada apenas quando testar Expert Advisor e está destinada sobretudo para o cálculo de um valor que é usado como um critério máximo personalizado para otimização de parâmetros de entrada genérica.
- TesterInit - este evento é gerado durante o início da otimização no Provador de Estratégia e antes do primeiro passo. O evento TesterInit é tratado usando a função OnTesterInit(). Durante o início da otimização, um Expert Advisor com este handler é carregado automaticamente em um gráfico de terminal separado com o ativo e período especificados no testador, e recebe o evento TesterInit. Esta função é usada para inicializar um Expert Advisor antes de iniciar a otimização para posterior processamento dos resultados da otimização.
- TesterPass - este evento é gerado quanto um novo data frame é recebido. O evento TesterPass é tratado usando a função OnTesterPass(). Um Expert Advsor com este handler é carregado automaticamente em um gráfico de terminal separado com o ativo/período especificados para teste, e recebe um evento TesterPass quando um frame é recebido durante otimização. A função é usado para tratamento dinâmico de resultados de otimização "in loco" sem esperar pela sua conclusão. Frames são adicionados usando a função FrameAdd(), que pode ser chamada após o fim de um passo individual no handler OnTester().
- TesterDeinit - este evento é gerado após o fim da otimização do Expert Advisor no Provador de Estratégia. O evento TesterDeinit é tratado usando a função OnTesterDeinit(). Um Expert Advisor com este handler é carregado automaticamente em um gráfico no começo da otimização, e recebe TesterDeInit após sua conclusão. A função é usado para o processamento final de todos os resultados da otimização.

Agentes de Teste

O teste no terminal cliente MetaTrader 5 é realizado usando agentes de teste. Agentes locais são criados e habilitados automaticamente. O número padrão de agentes locais corresponde ao número de núcleos em um computador.

Cada agente de teste tem sua própria cópia das variáveis globais, que não são relacionados com o terminal cliente. O terminal em si é o despachante, que distribui as tarefas para os agentes locais e

© 2000-2024, MetaQuotes Ltd.

remotos. Apos executar uma tarefa no teste de um EA, com os parâmetros dados, o agente retorna o resultado para o terminal. Para um teste individual, somente um agente é usado.

O agente armazena o histórico, recebido do terminal, em pastas separadas, pelo nome do instrumento, assim o histórico para EURUSD é armazenado em uma pasta chamada EURUSD. Além disso, o histórico dos instrumentos é separado pelas suas fontes. A estrutura para armazenar o histórico tem o seguinte forma:

```
tester_catalog\Agent-IPaddress-Port\bases\name_source\history\symbol_name
```

Por exemplo, o histórico para EURUSD proveniente do servidor MetaQuotes-Demo pode ser armazenada na pasta tester_catalog\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD.

Um agente local, após a conclusão de um teste, entra em modo de espera, aguardando a próxima tarefa por mais 5 minutos, de modo a não perder tempo com o lançamento para a próxima chamada. Somente após o período de espera acabar, o agente local desliga e se descarrega da memória do CPU.

No caso de uma conclusão antecipada do testador, por parte do usuário (o botão "Cancelar"), bem como com o fechamento do terminal cliente, todos os agentes locais imediatamente param seu trabalhar e são descarregados da memória.

A Troca de Dados entre o Terminal e o Agente

Ao rodar um teste, o terminal cliente prepara para enviar ao agente um número de blocos de parâmetros:

- Parâmetros de input para o teste (modo de simulação, o intervalo do teste, instrumentos, critério de otimização, etc.)
- A lista dos ativos selecionados na janela "Observação de Mercado"
- A especificação do ativo do teste (o tamanho do contrato, as margens permitidas do mercado para definir um StopLoss e Takeprofit, etc)
- O Expert Advisor (robot) para ser testado e os valores de seus parâmetros de entrada
- Informações sobre arquivos adicionais (bibliotecas, indicadores, arquivos de dados - # property tester_ ...)

tester_indicator	string	Nome de um indicador customizado no formato de "indicator_name.ex5". Indicadores que requerem teste são definidos automaticamente a partir da chamada à função iCustom(), se o correspondente parâmetro estiver definido através de um string constante. Para todos os outros casos (use da função IndicatorCreate() ou uso de uma string não constante no parâmetro que defini o nome do indicador) esta propriedade é requerida
tester_file	string	Nome de arquivo para um testador com a indicação de extensão, entre aspas duplas (como uma string constante). O arquivo especificado será passado para o Testador de Estratégias. Arquivos de entrada para serem testados, se forem necessários, devem sempre ser especificados.

© 2000-2024, MetaQuotes Ltd.

tester_indicator	string	Nome de um indicador customizado no formato de "indicator_name.ex5". Indicadores que requerem teste são definidos automaticamente a partir da chamada à função iCustom(), se o correspondente parâmetro estiver definido através de um string constante. Para todos os outros casos (use da função IndicatorCreate() ou uso de uma string não constante no parâmetro que defini o nome do indicador) esta propriedade é requerida
tester_library	string	Nome da biblioteca (library) com a extensão, entre aspas duplas. Uma biblioteca pode ter extensão dll ou ex5. Bibliotecas que requerem teste são definidas automaticamente. No entanto, se alguma das bibliotecas é usada por um indicador customizado, esta propriedade é requerida

Para cada bloco de parâmetros, uma impressão digital na forma de MD5-hash é criada, que é enviado para o agente. O MD5-hash é único para cada conjunto, seu volume é muito menor que a quantidade de informação na qual ele é calculado

O agente recebe um hash de blocos e os compara com aqueles que ele já tem. Se a impressão digital do bloco de parâmetros dado não está presente no agente, ou o hash recebido é diferente daquele existente, o agente solicita este bloco de parâmetros. Isso reduz o tráfego entre o terminal e o agente.

Após o teste, o agente retorna ao terminal todos os resultados da execução, que são exibidos nas guias "Resultados" e "Resultados da Otimização": o lucro recebido, o número de operações (deals), o coeficiente Sharpe, o resultdo da função OnTester(), etc.

Durante a otimização, o terminal distribui tarefas de teste para os agentes em pequenos pacotes, cada pacote contendo várias tarefas (cada tarefa significando testes individuais com um conjunto de parâmetros de entrada). Isso reduz o tempo de troca entre o terminal e o agente.

Os agentes nunca gravam no disco rígido o arquivo EX5, obtido do terminal (EA, indicadores, bibliotecas, etc.) for razões de segurança, de modo que um computador com um agente em execução não pode usar os dados enviados. Todos os outros arquivos, incluindo DLL, são gravados na caixa de areia. Em agentes remotos você não pode testar EAs usando DLL.

Os resultados do teste são adicionados pelo terminal em uma cache especial de resultados (o cache de resultado), para um acesso rápido a estes resultados quando necessário. Para cada conjunto de parâmetros, o terminal busca o cache de resultados por resultados já disponíveis a partir de execuções anteriores, a fim de evitar re-execuções. Se o resultado com tal conjunto de parâmetros não é encontrado, ao agente é dado a tarefa de conduzir o teste.

Todo tráfego entre o terminal e o agente é encripto-grafado.

Ticks não são enviados pela rede, eles são gerados nos agentes de teste.

Usando a Pasta Compartilhada para Todos os Terminais Cliente

Todos os agentes de teste são isolados uns dos outros e do terminal cliente: cada agente tem sua própria pasta na qual seus logs são gravados. Além disso, todas as operações de arquivo durante o

© 2000-2024, MetaQuotes Ltd.

teste do agente ocorrem na pasta **agent_name/MQL5/Files**. Contudo, nós podemos implementar a interação entre os agentes locais e o terminal cliente através de uma pasta compartilhada para todos os terminais clientes, se durante a abertura do arquivo você especificar a flag FILE_COMMON:

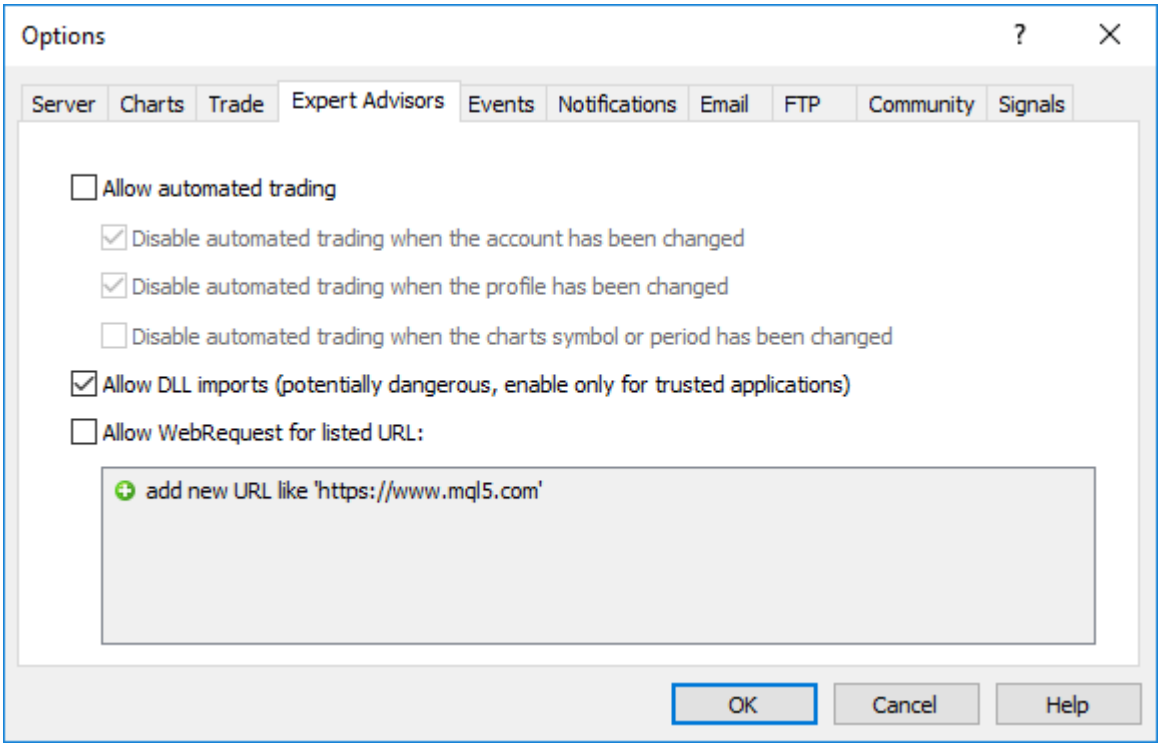
```
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- a pasta compartilhada para todos os terminais cliente
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- imprime o nome desta pasta
   PrintFormat("Abrir o arquivo na pasta compartilhada dos terminais de cliente %s", c
//--- abre um arquivo na pasta compartilhada (indicada pelo flag FILE_COMMON)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... novas ações
//---
   return(INIT_SUCCEEDED);
  }
```

Usando DLLs

Para acelerar a otimização nós podemos usar não somente agentes locais, mas também agentes remotos. Neste caso, existem algumas limitações para agentes remotos. Primeiramente, agentes remotos não exibem em seus logs os resultados da execução da função Print(), mensagens sobre a abertura e enceramento de posições. Um mínimo de informação é exibida no log para impedir que EAs escritos incorretamente de atolem o computador, no qual o agente remoto está trabalhando, com mensagens.

A segunda limitação - a proibição no uso de DLL ao testar EAs. Chamadas de DLL são absolutamente proibidas em agentes remotos por razões de segurança. Em agentes locais, chamadas de DLL em EAs testados são permitidas somente com a apropriada permissão "Allow import DLL".

© 2000-2024, MetaQuotes Ltd.



Observação: Ao usar os códigos recebidos dos EAs (scripts, indicadores) que requerem a permissão para fazer chamadas de DLL, você deve estar ciente dos riscos, que você assume ao permitir este opção nas configurações do terminal. Independentemente de como a EA será usada - para teste ou para execução em um gráfico.

© 2000-2024, MetaQuotes Ltd.

As variáveis predefinidas

Para cada programa MQL5 executável, um conjunto de variáveis predefinidas é suportado, que reflete o estado do gráfico de preço corrente no momento em que um programa-mql5 é iniciado.

Os valores das variáveis predefinidas são definidas pelo terminal cliente antes de um programa MQL5 ser iniciado. Variáveis predefinidas são constantes e não podem ser alteradas a partir de um programa MQL5. Como exceção, existe a variáveis especial _LastError, que pode ser redefinida para 0 pela função ResetLastError.

Variável	Valor
_AppliedTo	A variável _AppliedTo permite ver no indicador o tipo de dados em que é calculado
_Digits	Número de casas decimais
_Point	Tamanho do ponto do ativo corrente na moeda de cotação
_LastError	O último código de erro
_Period	Janela de tempo do gráfico corrente
_RandomSeed	Status atual do gerador de números pseudo-aleatórios
_StopFlag	Flag de parada de programa
_Symbol	Nome do ativo do gráfico corrente
_UninitReason	Código do motivo da não inicialização
_IsX64	A variável _IsX64 permite saber em que terminal está funcionando o programa MQL5

Variáveis predefinidas não podem ser definidas em uma biblioteca. Uma biblioteca usa tais variáveis que são definidas em um programa a partir do qual esta biblioteca é chamada.

© 2000-2024, MetaQuotes Ltd.

int _AppliedTo

A variável _AppliedTo permite ver no indicador o tipo de dados em que é calculado:

Tipo de dados	Valor	Descrição dos dados usados para calcular o indicador
—	0	O indicador usa a segunda maneira de chamada de OnCalculate() - os dados para o cálculo não são especificados por um buffer ou array de dados
Close	1	Preço Close
Open	2	Preço Open
High	3	Preço High
Low	4	Preço Low
Median Price (HL/2)	5	Preço médio = (High+Low)/2
Typical Price (HLC/3)	6	Preço típico = (High+Low+Close)/3
Weighted Price (HLCC/4)	7	Preço ponderado = (Open+High+Low+Close)/4
Previous Indicator's Data	8	Dados do indicador inicializado no gráfico antes deste indicador
First Indicator's Data	9	Dados do indicador inicializado no gráfico pela primeira vez
Indicator handle	10+	Dados do indicador transferidos para a função iCustom() com ajuda do identificador do indicador. O valor _AppliedTo contém o identificador do indicador

Exemplo:

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- indicator buffers mapping
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// obtemos o tipo de dados no qual o indicador é calculado
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Descrição dos dados em que o indicador é calculado                |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="It's first type of OnCalculate() - no data buffer";
         break;
      case(1):descr="Indicator calculates on Close price";
         break;
      case(2):descr="Indicator calculates on Open price";
         break;
      case(3):descr="Indicator calculates on High price";
         break;
      case(4):descr="Indicator calculates on Low price";
         break;
      case(5):descr="Indicator calculates on Median Price (HL/2)";
         break;
      case(6):descr="Indicator calculates on Typical Price (HLC/3)";
         break;
      case(7):descr="Indicator calculates on Weighted Price (HLCC/4)";
         break;
      case(8):descr="Indicator calculates Previous Indicator's data";
         break;
      case(9):descr="Indicator calculates on First Indicator's data";
         break;
      default: descr="Indicator calculates on data of indicator with handle="+string(c
         break;
     }
//---
   return descr;
  }
```

Veja também

ENUM_APPLIED_PRICE

© 2000-2024, MetaQuotes Ltd.

int _Digits

A variável _Digits armazena o número de dígitos após o ponto decimal, que define a precisão do preço do ativo do gráfico corrente.

Você pode também usar a função Digits().

© 2000-2024, MetaQuotes Ltd.

double _Point

A variável _Point contém o tamanho do ponto do ativo corrente na moeda de cotação.

Você também pode usar a função Point().

© 2000-2024, MetaQuotes Ltd.

int _LastError

A variável _LastError contém o código do último erro, que ocorreu durante a execução de um programa-mql5. Seu valor pode ser redefinido para zero por ResetLastError().

Para obter o código do último erro, você também pode usar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

A variável _Period contém o valor da janela de tempo do gráfico corrente.

Você também pode usar a função Period().

Também Veja

PeriodSeconds, Janelas de tempo de gráficos, Data e Hora, Visibilidade de objetos

© 2000-2024, MetaQuotes Ltd.

_RandomSeed

Variável para armazenamento do estado corrente ao gerar inteiros pseudo-aleatórios. _RandomSeed altera seu valor ao chamar MathRand(). Use MathSrand() para definir a condição inicial requerida.

x número aleatório recebido pela função MathRand() é calculado da seguinte forma a cada chamada:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Também Veja

MathRand(), MathSrand(), Tipos inteiros

© 2000-2024, MetaQuotes Ltd.

bool _StopFlag

A variável _StopFlag contém o flag de parada de um programa MQL5. Quando o terminal cliente está tentando parar o programa, ele define a variável _StopFlag para true.

Para verificar o estado do _StopFlag você também pode usar a função IsStopped().

© 2000-2024, MetaQuotes Ltd.

string _Symbol

A variável _Symbol contém o nome do ativo do gráfico corrente.

Você também pode usar a função Symbol().

© 2000-2024, MetaQuotes Ltd.

int _UninitReason

A variável _UninitReason contém o código do motivo de não inicialização do programa.

Geralmente, este código é obtido através da função UninitializeReason().

© 2000-2024, MetaQuotes Ltd.

int _IsX64

A variável _IsX64 permite saber em que terminal está funcionando o programa MQL5: para um terminal de 32 bits _IsX64=0 e para um terminal de 64 _IsX64!=0.

Você também pode usar a função TerminalInfoInteger(TERMINAL_X64).

Exemplo:

```
// verifique em que terminal está sendo executado o programa
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("O programa ",__FILE__," está sendo executado em um terminal de 64 bits");
   else
      Print("O programa ",__FILE__," está sendo executado em um terminal de 32 bits");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Veja também

MQLInfoInteger, Importando funções (#import)

© 2000-2024, MetaQuotes Ltd.

Funções Comuns

Funções de propósito geral não inclusas em nenhum grupo especializado estão listadas aqui.

Função	Ação
Alert	Exibe uma mensagem em uma janela separada
CheckPointer	Retorna o tipo do ponteiro de objeto
Comment	Gera um comentário no canto superior esquerdo do gráfico
CryptEncode	Transforma os dados do array com o método especificado
CryptDecode	Executa a transformação inversa dos dados do array
DebugBreak	Ponto de parada de programa em depuração
ExpertRemove	Interrompe um Expert Advisor e o descarrega do gráfico
GetPointer	Retorna o ponteiro de objeto
GetTickCount	Retorno o número de milissegundos decorridos desde que o sistema foi iniciado
GetTickCount64	Retorno o número de milissegundos decorridos desde que o sistema foi iniciado
GetMicrosecondCount	Retorna o número de microssegundos que decorreram desde o início do programa MQL5
MessageBox	Cria, exibe uma caixa de mensagem e o gerencia
PeriodSeconds	Retorna o número de segundos no período
PlaySound	Toca um arquivo de som
Print	Exibe uma mensagem no log
PrintFormat	Formata e imprime um conjunto de símbolos e valores em um arquivo de log de acordo com um formato predefinido
ResetLastError	Define o valor da variável predeterminada _LastError para zero
ResourceCreate	Cria um recurso de imagem baseado em um conjunto de dados
ResourceFree	Exclui recursos criados dinamicamente (liberando a memória alocada para ele)
ResourceReadImage	Lê os dados do recurso gráfico criado pela função ResourceCreate() ou salvo em arquivo EX5 durante a compilação
ResourceSave	Salva um recurso em um arquivo especificado
SetUserError	Define a variável predefinida _LastError para o valor igual a ERR_USER_ERROR_FIRST + user_error
SetReturnError	Define o código de retorno que devolverá um processo do terminal ao encerrar o trabalho

© 2000-2024, MetaQuotes Ltd.

Função	Ação
Sleep	Suspende por um intervalo especificado a execução do Expert Advisor corrente ou script
TerminalClose	Ordena o terminal a finalizar a operação
TesterHideIndicators	Define o modo mostrar/ocultar indicadores usados no Expert Advisor
TesterStatistics	Retorna o valor de uma estatística específica calculada com base em resultados de teste
TesterStop	Dá o comando para encerrar o programa durante os testes.
TesterDeposit	Função especial para emular operações de depósito durante o teste. Pode ser usado em alguns sistemas de gerenciamento de capital
TesterWithdrawal	Função especial para emular a operação de saque de dinheiro no processo de teste
TranslateKey	Retorna o caractere Unicode de acordo com o código de tecla virtual
ZeroMemory	Redefine uma variável passada para ela por referência. A variável pode ser de qualquer tipo, exceto classes e estruturas que tenham construtores.

© 2000-2024, MetaQuotes Ltd.

Alert

Exibe uma mensagem em uma janela separada.

```
void  Alert(
   argument,      // primeiro valor
   ...            // outros valores
   );
```

Parâmetros

argument

[in] Quaisquer valores separados por virgulas. Para dividir a saída da informação em várias linhas você pode usar o caractere de quebra de linha "\n" ou "\r\n". O número de parâmetros não pode exceder 64.

Valor do Retorno

Sem valor de retorno.

Observação

Arrays não podem ser passados para a função Alert(). Arrays devem ser fornecidos elemento a elemento. Dados do tipo double são exibidos com 8 dígitos após o ponto decimal, dados do tipo float são exibidos com 5 dígitos após o ponto decimal. Para exibir os números reais com uma precisão diferente ou em notação científica, use a função DoubleToString().

Dados do tipo bool são exibidos como strings "true" ou "false". Datas são exibidas como YYYY.MM.DD HH:MM:SS. Para exibir uma data em um outro formato use a função TimeToString(). Dados do tipo color são exibidos tanto como uma string R,G,B ou como nome de cor, se a cor estiver presente no conjunto de cores.

Alert() function does not work in the Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

CheckPointer

A função retorna o tipo do ponteiro de objeto.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // ponteiro de objeto
   );
```

Parâmetros

anyobject

[in] Ponteiro de objeto.

Valor de retorno

Retorna um valor da enumeração ENUM_POINTER_TYPE.

Observação

Uma tentativa de chamar um ponteiro incorreto resulta na finalização crítica de um programa. É por isso que é necessário chamar a função CheckPointer antes de usar um ponteiro. Um ponteiro pode estar incorreto nos seguintes casos:

- o ponteiro é igual a NULL;
- o objeto foi excluído usando o operador delete.

Esta função pode ser usada para checar a validade do ponteiro. Um valor não zero garante que o ponteiro pode ser usado para acesso.

Para uma verificação rápida, você também pode usar o operador "!"(exemplo) que testa sua validade por meio de uma chamada de função implícita CheckPointer.

Exemplo:

```
//+------------------------------------------------------------------+
//| Exclui lista através da exclusão de seus elementos               |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- ponteiro de serviço para trabalhar no loop
   CItem* item;
//--- percorre pelo loop e tenta excluir ponteiros dinâmicos
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," para ser excluído");
         delete (item);
        }
      else Print("Objeto não-dinâmico ",item.Identifier()," cannot be deleted");
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//---
  }
```

Também Veja

Ponteiros de Objetos, Verificação do Ponteiro de Objeto, Exclusão de Objeto Operador delete

© 2000-2024, MetaQuotes Ltd.

Comment

Esta função exibe um comentário definido pelo usuário no canto superior esquerdo de um gráfico.

```
void  Comment(
   argument,      // primeiro valor
   ...            // próximos valores
   );
```

Parâmetros

...

[in] Quaisquer valores, separados por vírgulas. Para delimitar a exibição da informação em várias linhas, um símbolo de quebra de linha "\n" ou "\r\n" é usado. O número de parâmetros não pode exceder 64. O comprimento total do comentário de entrada (incluindo símbolos invisíveis) não pode exceder 2045 caracteres (símbolos em excesso serão eliminados da exibição)

Valor do Retorno

Sem valor de retorno

Observação

Arrays não podem ser passados para a função Comment(). Arrays devem ser alimentados elemento a elemento.

Dados de tipo double são impressos com precisão de até 16 dígitos após o ponto decimal, e podem ser impressos tanto na forma tradicional quanto na forma científica, dependendo de qual notação será a mais compacta. Dados do tipo float são exibidos com 5 dígitos após o ponto decimal. Para exibir números reais com uma outra precisão ou em um formato predefinido, use a função DoubleToString().

Dados de tipo bool são impressos como strings "true" ou "false". Datas são mostradas como YYYY.MM.DD HH:MM:SS. Para mostrar datas em um outro formato, use a função TimeToString(). Dados de tipo color são impressos tanto como strings R,G,B quanto como nome de cor, se esta cor estiver presente no conjunto de cores.

A função Comment() não funciona no durante a otimização no Testador de Estratégias.

Exemplo:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Exibe valores em 3 linhas
   Comment(StringFormat("Mostrar preços\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spre
  }
```

Também Veja

© 2000-2024, MetaQuotes Ltd.

ChartSetString, ChartGetString

© 2000-2024, MetaQuotes Ltd.

CryptEncode

Transforma os dados do array com o método especificado.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // método
   const uchar&        data[],         // array fonte
   const uchar&        key[],          // key
   uchar&              result[]        // array de destino
   );
```

Parâmetros

método

[in] Método de transformação de dados. Pode ser um dos valores da enumeração ENUM_CRYPT_METHOD.

data[]

[in] Array fonte.

key[]

[in] Key array.

result[]

[out] Array de destino.

Valor retornado

Quantidade de bytes no array de destino ou 0 em caso de erro. Para obter informações sobre o erro chame a função GetLastError().

Exemplo:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- verificar
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- transformar em HEX string
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   string text="A rápida raposa marrom salta sobre o cão preguiçoso";
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- preparar key
   StringToCharArray(keystr,key);
//--- copiar texto do array fonte src[]
   StringToCharArray(text,src);
//--- imprimir dados iniciais
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s
//--- encriptar src[] com a key DES 56-bit em key[]
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- verificar erro
   if(res>0)
     {
      //--- imprimir dados encriptados
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- decodificar dst[] para src[]
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- verificar erro
      if(res>0)
        {
         //--- imprimir dados decodificados
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("Erro no CryptDecode. Error code=",GetLastError());
     }
   else
      Print("Erro no CryptEncode. Error code=",GetLastError());
  }
```

Veja também

Funções do array, CryptDecode()

© 2000-2024, MetaQuotes Ltd.

CryptDecode

Executa a transformação inversa dos dados do array, transformada pelo CryptEncode().

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,         // método
   const uchar&        data[],         // array fonte
   const uchar&        key[],          // key
   uchar&              result[]        // array de destino
   );
```

Parâmetros

método

[in] Método de transformação de dados. Pode ser um dos valores da enumeração ENUM_CRYPT_METHOD.

data[]

[in] Array fonte.

key[]

[in] Key array.

result[]

[out] Array de destino.

Valor retornado

Quantidade de bytes no array de destino ou 0 em caso de erro. Para obter informações sobre o erro chame a função GetLastError().

Veja também

Funções do array, CryptEncode()

© 2000-2024, MetaQuotes Ltd.

DebugBreak

É um ponto de parada de programa em depuração.

```
void  DebugBreak();
```

Valor do Retorno

Sem valor de retorno.

Observação

A execução de um programa MQL5 é interrompido somente se ele for iniciado em modo de depuração. A função pode ser usada para ver valores de variáveis e/ou para avançar a execução passo a passo.

© 2000-2024, MetaQuotes Ltd.

ExpertRemove

A função interrompe um Expert Advisor e o descarrega de um gráfico.

```
void  ExpertRemove();
```

Valor de retorno

Sem valor de retorno.

Observação

O Expert Advisor não é interrompido imediatamente ao chamar ExpertRemove(); apenas um flag para interromper a operação o EA é definido. Isso é, nenhum evento seguinte será processo, OnDeinit() será chamado e o Expert Advisor será descarregado e removido do gráfico.

A chamada da ExpertRemove() no testador de estratégias dentro do manipulador OnInit() faz com que seja cancelado o teste no conjunto de parâmetros atual. Essa finalização é considerada um erro durante a inicialização.

Ao chamar a ExpertRemove() no testador de estratégias após a inicialização bem-sucedida do EA, o teste é encerrado de maneira geral com chamada da OnDeinit() e da OnTester(). Neste caso, serão obtidos todas as estatísticas de negociação e valores do critério de otimização.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Test_ExpertRemove.mq5 |
//|                       Copyright 2009, MetaQuotes Software Corp. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// número de ticks antes do EA descarregar
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," código de motivo = ",reason);
//--- "limpa" comentário
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
```

```
//---
   tick_counter++;
   Comment("\nAntes de descarregar Expert Advisor ",__FILE__," left",
           (ticks_to_close-tick_counter)," ticks");
//--- antes
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," Expert Advisor será descarregado");
     }
   Print("tick_counter =",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

Também Veja

Execução de programas, Eventos do terminal Cliente

© 2000-2024, MetaQuotes Ltd.

GetPointer

A função retorna o ponteiro de objeto.

```
void*  GetPointer(
   any_class anyobject      // objeto de qualquer classe
   );
```

Parâmetros

anyobject

[in] Objeto de qualquer classe.

Valor do Retorno

A função retorna o ponteiro de objeto.

Observação

Somente objetos de classe têm ponteiros. Instâncias de estruturas e variáveis de tipo simples não podem ter ponteiros. O objeto de classe não criado usando o operador new(), mas, por exemplo, criado automaticamente no array de objetos, tem um ponteiro. Mas este ponteiro será do tipo automático POINTER_AUTOMATIC, portanto o operador delete() não pode ser aplicado a ele. Além de que, o tipo ponteiro não difere dos ponteiros dinâmico do tipo POINTER_DYNAMIC.

Como variáveis de tipo estrutura e de tipo simples não têm ponteiros, é proibido aplicar a função GetPointer() a elas. É também proibido passar o ponteiro como um argumento de função. Nos outros casos o resultado é indefinido.

Uma tentativa de chamar um ponteiro incorreto causa a finalização crítica de um programa. É por isso que a função CheckPointer() deve ser chamada antes de usar um ponteiro. Um ponteiro pode estar incorreto nos seguintes casos:

- o ponteiro é igual a NULL;
- o objeto foi excluído usando o operador delete.

Esta função pode ser usada para verificar a validade de um ponteiro. Um valor não-zero garante que o ponteiro pode ser usado para acesso.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              Check_GetPointer.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| Classe implementando o elemento da lista                         |
//+------------------------------------------------------------------+
```

```
class CItem
  {
   int                m_id;
   string             m_comment;
   CItem*             m_next;
public:
                      CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                     ~CItem() { Print("Destrutor de ",m_id,
                                      (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                      "dynamic":"não dinâmico"); }
   void               Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void               PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int                Identifier() { return(m_id); }
   CItem*             Next() {return(m_next); }
   void               Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//| Classe mais simples de lista                                     |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*             m_items;
public:
                      CMyList() { m_items=NULL; }
                     ~CMyList() { Destroy(); }
    bool              InsertToBegin(CItem* item);
    void              Destroy();
  };
//+------------------------------------------------------------------+
//| Incluindo elemento de lista no começo                            |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Excluindo a lista através da exclusão dos elementos              |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- ponteiro de serviço para trabalhar em um loop
   CItem* item;
//--- varre o loop e tenta excluir ponteiros dinâmicos
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dynamyc object ",item.Identifier()," para ser excluído");
          delete (item);
         }
       else Print("Objeto não-dinâmico ",item.Identifier()," cannot be deleted");
      }
//---
  }
//+------------------------------------------------------------------+
//| Função start do programa script                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- cria e adiciona na lista um ponteiro de objeto dinâmico
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dinâmico");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- adiciona ponteiros automáticos na lista
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automático");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- adiciona mais um ponteiro de objeto dinâmico no começo da lista
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dinâmico");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- exclui todos os elementos da lista
   list.Destroy();
//--- todos os elementos da lista será excluídos após o script terminar
//--- veja a guia Experts no terminal
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

Também Veja

Ponteiros de Objetos, Verificação do Ponteiro de Objeto, Exclusão de Objeto Operador delete

© 2000-2024, MetaQuotes Ltd.

GetTickCount

A função GetTickCount() retorna o número de milissegundos decorridos desde o inicialização do sistema.

```
uint  GetTickCount();
```

Valor do Retorno

Valor de tipo uint.

Observação

O contador é limitado pelas restrições do timer do sistema. O tempo é armazenado como um inteiro sem sinal, assim ele é sobre-preenchido a cada 49.7 dias se um computador trabalhar ininterruptamente.

Exemplo:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//| Script para medir o tempo de computação de 40 membros Fibonacci  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Lembra o valor inicial
   uint start=GetTickCount();
//--- Uma variável para obtenção do próximo número na série Fibonacci
   long fib=0;
//--- No loop calcular a quantidade específica de números de série de Fibonacci
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- Obtém o tempo gasto em milissegundos
   uint time=GetTickCount()-start;
//--- Imprime uma mensagem para o diário Experts
   PrintFormat("Calculando %d primeiros números Fibonacci tiveram %d ms",MAX_SIZE,time
//--- Script concluído
   return;
  }
//+------------------------------------------------------------------+
//| Função para obter número de Fibonacci pelo seu número serial     |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- O primeiro membro da série Fibonacci
   if(n<2) return(1);
//--- Todos os outros membros são calculados pela seguinte fórmula
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Também Veja

Data e Hora, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2024, MetaQuotes Ltd.

GetTickCount64

A função GetTickCount() 64 retorna o número de milissegundos que passaram desde o início do sistema.

```
ulong  GetTickCount64();
```

Valor retornado

Valor do tipo ulong.

Observação

O contador é limitado pela resolução do temporizador do sistema, que geralmente retorna um resultado com uma precisão de 10 a 16 milissegundos. Ao contrário da função GetTickCount, que é do tipo uint e, consequentemente, é abarrotada a cada 49,7 dias durante operação contínua do computador, GetTickCount64() pode ser usada com período de operação indefinido do computador e sem receio de abarrotamento.

Veja também

Data e hora, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

© 2000-2024, MetaQuotes Ltd.

GetMicrosecondCount

A função GetMicrosecondCount() retorna o número de microssegundos que decorreram desde o início do programa MQL5.

```
ulong  GetMicrosecondCount();
```

Retornar Valor

Valor do tipo ulong.

Exemplo:

```
//+------------------------------------------------------------------+
//| Teste da função                                                  |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- número de medidas
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- selecionar tipo de medida
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- executar código
         Test();
         //--- adicionar resultado da medida (dependendo do tipo)
```

© 2000-2024, MetaQuotes Ltd.

```
          if(n==0) ui_res+=GetTickCount()-ui;
          else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- Calcular o tempo mínimo e máximo para ambas as medições
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-ul_min)/1000.0,2))
  }
```

Veja também

Data e hora, GetTickCount, GetTickCount64

© 2000-2024, MetaQuotes Ltd.

MessageBox

Cria e exibe uma caixa de mensagem e o gerencia. Um caixa de mensagem contém uma mensagem e cabeçalho, e qualquer combinação de sinais predefinidos e botões de comando.

```
int  MessageBox(
   string  text,              // texto da mensagem
   string  caption=NULL,      // cabeçalho da caixa
   int     flags=0            // define o conjunto de botões na caixa
   );
```

Parâmetros

text

[in] Texto, contém a mensagem de saída.

caption=NULL

[in] Texto opcional a ser exibido no cabeçalho da caixa. Se o parâmetro estiver vazio, o nome do Expert Advisor é exibido no cabeçalho da caixa.

flags=0

[in] Flags opcionais definindo a aparência e comportamento da caixa de mensagem. Flags podem ser uma combinação de um grupo especial de flags.

Valor do Retorno

Se a função for executada com sucesso, o valor de retorno é um dos valores dos códigos de retorno do MessageBox().

Observação

A função é proibida em indicadores personalizados, uma vez que a chamada para MessageBox() interrompe o trabalho do thread de execução durante todo o tempo de espera pela resposta do usuário. Como todos os indicadores de cada símbolo são executados num único thread, o funcionamento de todos os gráficos em todos os períodos gráficos do símbolo em questão será interrompido.

MessageBox() function does not work in the Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

PeriodSeconds

Esta função retorna o número de segundos em um período.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // período do gráfico
   );
```

Parâmetros

period=PERIOD_CURRENT

[in] Valor do período de um gráfico proveniente da enumeração ENUM_TIMEFRAMES. Se o parâmetro não for especificado, ela retorna o número de segundos do período do gráfico corrente, no qual o programa está rodando.

Valor do Retorno

Número de segundos em um período selecionado.

Também Veja

_Period, Janelas de tempo de Gráficos, Date e Hora, Visibilidade de objetos

© 2000-2024, MetaQuotes Ltd.

PlaySound

Toca um arquivo de som.

```
bool  PlaySound(
   string  filename      // nome do arquivo
   );
```

Parâmetros

filename

[in] Caminho para um arquivo de som. Se filename=NULL, a reprodução é interrompida.

Valor do Retorno

true – se o arquivo for encontrado, caso contrário - false.

Observação

O arquivo deve estar localizado em terminal_directory\Sounds ou em sub-diretórios. Somente arquivos WAV são executados.

Chamar o PlaySound() com o parâmetro NULL interrompe a reprodução.

A função PlaySound() não funciona no Testador de Estratégias.

Também Veja

Recursos

© 2000-2024, MetaQuotes Ltd.

Print

Entra uma mensagem no log do Expert Advisor. Parâmetros podem ser de qualquer tipo.

```
void  Print(
   argument,      // primeiro valor
   ...            // próximos valores
   );
```

Parâmetros

...

[in] Quaisquer valores separados por virgulas. O número de parâmetros não pode exceder 64.

Observação

Arrays não podem ser passados para a função Print(). Arrays devem ser passados elemento a elemento.

Dados do tipo double são mostrados com precisão de até 16 dígitos após o ponto decimal, e podem ser exibidos tanto na formato tradicional como em notação científica, dependendo de qual saída será mais compacta. Dados do tipo float são exibidos com 5 dígitos após o ponto decimal. Para imprimir números reais com uma outra precisão ou em um formato predefinido, use a função PrintFormat() .

Dados do tipo bool são exibidos como strings "true" ou "false". Datas são mostradas como YYYY.MM.DD HH:MM:SS. Para mostrar datas em um outro formato, use TimeToString(). Dados do tipo cor são retornados como string R,G,B ou como nome de cor, se esta cor estiver presente no conjunto de cores

A função Print() não funciona no durante a otimização no Testador de Estratégias.

Exemplo:

```
void OnStart()
  {
//--- Imprime o DBL_MAX usando Print(), isso é equivalente a PrintFormat(%%.16G,DBL_MA
   Print("---- como DBL_MAX aparece -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- Agora imprime o número DBL_MAX usando PrintFormat()
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Imprime no diário Experts
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- Veja como um float é impresso
   float c=(float)M_PI; // Nós devemos explicitamente converter para o tipo desejado
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- Mostra o que pode acontecer com operações aritméticas com tipos reais
   double a=7,b=200;
   Print("---- Antes de operações aritméticas");
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Dividir a por b (7/200)
   a=a/b;
//--- Agora emula a restauração de um valor na variável b
   b=7.0/a; // É esperado que b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - mas ele difere
//--- Imprimi o novo valor calculado de b
   Print("----- Após as operações aritméticas");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Imprime no diário Experts
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (veja que b não é mais igual a 20

//--- Cria um valor muito pequeno epsilon=1E-013
   double epsilon=1e-13;
   Print("---- Criando um valor muito pequeno");
   Print("epsilon=",epsilon); // Obter epsilon=1E-013
//--- Agora subtrai epsilon de b e novamente imprime o valor no diário Experts
   b=b-epsilon;
//--- Duas formas de uso
   Print("---- Depois de subtrair epsilon da variável b");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Imprime no diário Experts
// Print(b)=199.9999999999999  (agora o valor de b após subtrair epsilon não pode ser
// Print(DoubleToString(b,16))=199.9999999999998578
//    (agora o valor de b após subtrair epsilon não podem ser arredondado para 200)
  }
```

Também Veja

DoubleToString, StringFormat

© 2000-2024, MetaQuotes Ltd.

PrintFormat

Formata e entra conjuntos de símbolos e valores no log do Expert Advisor em concordância com um formato predefinido.

```
void  PrintFormat(
   string format_string,   // string de formato
   ...                     // valores de tipo simples
   );
```

Parâmetros

format_string

[in] A string de formato consiste de símbolos simples, e se a string de formato é seguido por argumentos, ela também contém especificações de formato.

...

[in] Quaisquer valores de tipos simples separados por vírgula. Número total de parâmetros não pode exceder A 64, incluindo a string de formato.

Valor do Retorno

String.

Observação

A função Printformat() não funciona no durante a otimização no Testador de Estratégias.

O número, ordem e tipo dos parâmetros devem corresponder exatamente com o conjunto de qualificadores, caso contrário, o resultado do print é indefinido. Ao invés de PrintFormat(), você pode usar printf().

Se a string de formato for seguida por parâmetros, esta string deve conter especificações de formato que denotem o formato de saída destes parâmetros. Especificações de formato sempre começam com o sinal de percentual (%).

Uma string de formato é lida da esquerda para direita. Quando a primeira especificação de formato é encontrada (se houver alguma), o valor do primeiro parâmetro após a string de formato é transformado e impresso de acordo com a especificação predefinida. A segunda especificação de formato chama a transformação e impressão do segundo parâmetro, e assim por diante até a string de formato acabe.

A especificação de formato tem a seguinte forma:

%[flags][largura][.precisão][{h | l | ll | I32 | I64}]tipo

Cada campo da especificação de formato é tanto um símbolo simples, como um número denotando uma opção de formato simples. A especificação de formato mais simples contém somente o sinal de percentagem (%) e um símbolo definindo o tipo do parâmetro de saída (por exemplo, %s). Se você precisar imprimir o sinal de percentual na string de formato, use o especificação de formato %%.

flags

© 2000-2024, MetaQuotes Ltd.

Flag	Descrição	Comportamento Default
- (menos)	Justificação a esquerda dentro da largura (width) predefinida	Justificação a direita
+ (mais)	Imprime o sinal + ou - de valores de tipos com sinal	O sinal somente é mostrada se o valor for negativo
0 (zero)	Zeros são adicionados antes de um valor impresso com a largura predefinida. Se o flag **0** for especificado com um formato inteiro (**i**, **u**, **x**, **X**, **o**, **d**) e a especificação de precisão for definida (por exemplo, %04.d), então 0 é ignorado.	Nada é adicionado
espaço	Um espaço é exibido antes de um valor impresso, se ele for um valor com sinal e positivo	Espaços não são inseridos
#	Se usado em conjunto com o formato **o**, **x** ou **X**, então antes do valor de saída 0, 0x ou 0X é adicionado respectivamente.	Nada é adicionado
	Se usado em conjunto com o formato **e**, **E**, **a** ou **A**, o valor é sempre mostrado com um ponto decimal.	Ponto decimal é mostrado somente se houver um parte fracionária não-zero.
	Se usado em conjunto com o formato **g** ou **G**, o flag define a presença de um ponto decimal no valor de saída e evita o truncamento de zeros à esquerda. O flag # é ignorado quando usado em conjunto com formatos **c**, **d**, **i**, **u**, **s**.	Ponto decimal é mostrado somente se houver um parte fracionária não-zero. Zeros à esquerda são cortados.

largura

Um número decimal não negativo que define o número mínimo de símbolos de saída do valor formatado. Se o número de saída de símbolos é menor do que a largura especificada, o correspondente número de espaços é adicionado a partir da esquerda ou direita, dependendo do alinhamento (flag -). Se houver o flag zero (0), o número correspondente de zeros é adicionado antes do valor de saída. Se o número de saída de símbolos de saída é maior do que a largura especificada, o valor de saída não é cortado.

Se um asterisco (*) for especificado como width, um valor de tipo int deve ser indicado no lugar correspondente da lista de parâmetros passados. Ele será usado para especificar a largura do valor de saída.

precisão

Um número decimal não negativo que define a precisão de saída - número de dígitos após o ponto decimal. Como diferença da especificação de largura, a especificação de precisão pode truncar a parte fracionário com ou sem arredondamento.

O uso da especificação de precisão é diferente para tipos de formato diferentes.

© 2000-2024, MetaQuotes Ltd.

Tipos	Descrição	Comportamento Default
a, A	Especificação de precisão define o número de dígitos após o ponto decimal.	Precisão padrão - 6.
c, C	Não usado	
d, i, u, o, x, X	Define o número mínimo de dígitos de saída. Se o número de dígitos em um parâmetro correspondente for menor que esta precisão, zeros são adicionados a esquerda do valor de saída. O valor de saída não é cortado, se o número de dígitos de saída é maior do que a precisão apurada.	Precisão padrão - 1.
e, E, f	Define o número de dígitos de saída após o ponto decimal. O último dígito é arredondado.	Precisão padrão - 6. Se a definição de precisão for 0 ou a parte decimal estiver ausente, o ponto decimal não é exibido.
g, G	Define o número máximo de números significativos.	6 números significativos são impressos.
s	Define o número de saída dos símbolos de uma string. Se o comprimento da string excede ao apurado, a string e cortada.	Toda a string é a saída.

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

Especificação de tamanhos dos dados, passados como parâmetro.

Tipo de Parâmetro	Prefixo Usado	Especificador Conjunto de Tipo
int	l (letra minúscula L)	d, i, o, x, or X
uint	l (letra minúscula L)	o, u, x, or X

© 2000-2024, MetaQuotes Ltd.

Tipo de Parâmetro	Prefixo Usado	Especificador Conjunto de Tipo
long	ll (duas letras minúsculas L)	d, i, o, x, or X
short	h	d, i, o, x, or X
ushort	h	o, u, x, or X
int	I32	d, i, o, x, or X
uint	I32	o, u, x, or X
long	I64	d, i, o, x, or X
ulong	I64	o, u, x, or X

tipo

Especificador de tipo é o único campo obrigatório para a saída formatada.

Símbolo	Tipo	Formato de Saída
c	int	Símbolo de tipo short (Unicode)
C	int	Símbolo de tipo char (ANSI)
d	int	Inteiro decimal com sinal
i	int	Inteiro decimal com sinal
o	int	Inteiro octal sem sinal
u	int	Inteiro decimal sem sinal
x	int	Inteiro hexadecimal sem sinal, usando "abcdef"
X	int	Inteiro hexadecimal sem sinal, usando "ABCDEF"
e	double	Um valor real no formato [-] d.ddde[sinal] ddd, onde d - um dígito decimal, dddd - um ou mais dígitos decimais, ddd - um número de três dígitos que determina o tamanho do exponente, sinal - mais ou menos
E	double	Semelhando ao formato de e, exceto que o sinal do expoente é impresso em letra maiúscula (**E** ao invés de e)
f	double	Um valor real no format [-] dddd.dddd, onde dddd - um ou mais dígitos decimais. Número de dígitos exibidos depois do ponto decimal depende do tamanho do valor do número. Número de dígitos após o ponto decimal depende da precisão requerida.
g	double	Uma saída de valor real no formato **f** ou **e**depende de qual saída é mais compacta.

© 2000-2024, MetaQuotes Ltd.

Símbolo	Tipo	Formato de Saída
G	double	Uma saída de valor real no formato **F** ou **E**depende de qual saída é mais compacta.
a	double	Um número real no formato [−]0xh.hhhh **p**±dd, onde h.hhhh – mantissa na forma de dígitos hexadecimais, usando "abcdef", dd - Um ou mais dígitos de expoente. Número de posições decimais é determinado pela especificação de precisão
A	double	Um número real no formato [−]0xh.hhhh **p**±dd, onde h.hhhh – mantissa na forma de dígitos hexadecimais, usando "abcdef", dd - Um ou mais dígitos de expoente. Número de posições decimais é determinado pela especificação de precisão
s	string	Saída da string

Ao invés de PrintFormat(), você pode usar printf().

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- trade server name
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- account number
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- long value output
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: leverage = 1:%I64d",
               server,login,leverage);
//--- account currency
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- double value output with 2 digits after the decimal point
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: account equity = %.2f %s",
               server,login,equity,currency);
//--- double value output with mandatory output of the +/- sign
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: current result for open positions = %+.2f %s",
               server,login,profit,currency);
//--- double value output with variable number of digits after the decimal point
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: point value  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: current spread in points = %d ",
               _Symbol,spread);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- output using PrintFormat() with default accuracy
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- simple output using Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* execution result
   MetaQuotes-Demo 1889998: leverage = 1:100
   MetaQuotes-Demo 1889998: account equity = 22139.86 USD
   MetaQuotes-Demo 1889998: current result for open positions = +174.00 USD
   EURUSD: point value  = 0.00001
   EURUSD: current spread in points = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

Também Veja

StringFormat, DoubleToString, Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

ResetLastError

Define o valor do variável predefinida _LastError para zero.

```
void  ResetLastError();
```

Valor do Retorno

Sem valor de retorno.

Observação

Deve ser notado que a função GetLastError() não zera a variável _LastError. Geralmente a função ResetLastError() é chamada antes de chamar uma função, após o qual um surgimento de erro é verificado.

© 2000-2024, MetaQuotes Ltd.

ResourceCreate

Criar um recurso de imagem baseado em um conjunto de dados. Existe duas variantes da função:

Criando um recurso baseado em um arquivo

```
bool  ResourceCreate(
   const string      resource_name,        // Nome do recurso
   const string      path                  // Uma caminho relativo para o arquivo
   );
```

Criando um recurso baseado em um array de pixels

```
bool  ResourceCreate(
   const string      resource_name,        // Nome do recurso
   const uint&       data[],               // Conjunto de dados em um array
   uint              img_width,            // A largura do recurso de imagem
   uint              img_height,           // A altura do recurso de imagem
   uint              data_xoffset,         // O deslocamento horizontal para direita d
   uint              data_yoffset,         // O deslocamento vertical para baixo do ca
   uint              data_width,           // A largura total da imagem baseada no con
   ENUM_COLOR_FORMAT color_format          // Método de processamento de cor
   );
```

Parâmetros

resource_name

[in] Nome do recurso.

data[][]

[in] Um array de uma dimensão ou duas dimensões de cores para criar uma imagem completa.

img_width

[in] A largura da área da imagem retangular em pixels para ser colocada no recurso na forma de uma imagem. Ela não pode ser maior que o valor *data_width*.

img_height

[in] A altura da área da imagem retangular em pixels para ser colocada no recurso na forma de uma imagem.

data_xoffset

[in] O deslocamento horizontal para direita da área retangular da imagem.

data_yoffset

[in] O deslocamento vertical para baixa da área retangular da imagem.

data_width

[in] Requerido somente para arrays de uma dimensão. Denota a largura total da imagem a partir do conjunto de dados. Se *data_width*=0, é assumido ser igual a *img_width*. Para arrays bidimensionais o parâmetro é ignorado e é assumido como sendo igual à segunda dimensão do array *data[]*.

color_format

© 2000-2024, MetaQuotes Ltd.

[in] Método de processamento de cor, a partir de um valor da enumeração ENUM_COLOR_FORMAT.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro chame a função GetLastError(). Os seguintes erros podem ocorrer:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (nomes idênticos do recurso dinâmico e estático)
- 4016 – ERR_RESOURCE_NOT_FOUND (o recurso não foi encontrado)
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (este tipo de recurso não é suportado)
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (o nome do recurso é muito longo)

Observação

Se a segunda versão da função é chamada para criar o mesmo recurso com largura, altura e parâmetros de deslocamento diferentes, ela não cria um novo recurso, mas simplesmente atualiza o recurso existente.

A primeira versão da função é usada para carregar imagens e sons a partir de arquivos, e a segunda versão é usada somente para a criação dinâmica de imagens.

Imagens devem estar no formato BMP com um profundidade de cor de 24 ou 32 bits. Sons podem somente ser no formato WAV. O tamanho do recurso não deve exceder 16 Mb.

ENUM_COLOR_FORMAT

Identifier	Descrição
COLOR_FORMAT_XRGB_NOALPHA	O componente do canal alfa é ignorado
COLOR_FORMAT_ARGB_RAW	Componentes de cor não são tratados pelo terminal (devem ser corretamente definidos pelo usuário)
COLOR_FORMAT_ARGB_NORMALIZE	Componentes de cor são tratados pelo terminal

Também Veja

Recursos, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceFree

A função exclui dinamicamente recursos criados (liberando a memória alocada para ele).

```
bool  ResourceFree(
   const string  resource_name      // nome do recurso
   );
```

Parâmetros

resource_name

[in] Nome do recurso deve começar com "::".

Valor de retorno

true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

Observação

ResourceFree() permite aos desenvolvedores de aplicativos MQL5 gerenciarem o consumo de memória quando se trabalha ativamente com recursos. Objetos gráficos vinculados ao recurso que será apagado da memória serão exibidos corretamente após a sua exclusão. No entanto, os objetos gráficos recém-criados ((OBJ_BITMAP e OBJ_BITMAP_LABEL) não poderão utilizar o recurso suprimido.

A função deleta apenas recursos dinâmicos criados pelo programa.

Também Veja

Recursos, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceReadImage

A função lê os dados do recurso gráfico criado pela função ResourceCreate() ou salvo como arquivo EX5 durante a compilação.

```
bool  ResourceReadImage(
   const string      resource_name,       // nome do recurso gráfico para leitura
   uint&             data[],              // array recebendo dados a partir do recurso
   uint&             width,               // para receber a largura da imagem no recu
   uint&             height,              // para receber a altura da imagem no recurs
   );
```

Parâmetros

resource_name

[in] Nome do recurso gráfico contendo uma imagem. Para acessar novamente seus próprios recursos, o nome é usado em formato abreviado "::resourcename". Se baixamos um recurso a partir de uma compilação de arquivo EX5, o nome completo seria usado com o caminho relativo do diretório MQL5, nomes do arquivo e recurso - "path\\filename.ex5::resourcename".

data[][]

[in] Array uni- ou bi-dimensional para recebimento de dados do recurso gráfico.

img_width

[out] Recurso gráfico da largura da imagem em pixels.

img_height

[out] Recurso gráfico da altura da imagem em pixels.

Valor de retorno

true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

Observação

Se é array *data[]*, então será usado para criar um recurso gráfico, formatos de cor COLOR_FORMAT_ARGB_NORMALIZE ou COLOR_FORMAT_XRGB_NOALPHA devem ser utilizados.

Se é array *data[]* bidimensional e sua segunda dimensão é menor do que X (largura) tamanho do recurso gráfico,a função ResourceReadImage() retorna falso e a leitura não é executada. Mas se o recurso existe, o tamanho da imagem real é retornada pelos parâmetros de largura e altura. Isto permitirá fazer outra tentativa para receber dados a partir do recurso.

Também Veja

Recursos, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

ResourceSave

Salva um recurso no arquivo especificado.

```
bool  ResourceSave(
   const string  resource_name      // Nome do recurso
   const string  file_name          // Nome do arquivo
   );
```

Parâmetros

resource_name

[in] O nome do recurso deve começar com "::".

file_name

[in] O nome do arquivo relativo a MQL5\Files.

Valor do Retorno

true - no caso de sucesso, caso contrário, false. Para informações do erro chamar GetLastError().

Observação

A função sempre sobre-escreve um arquivo e cria todos os diretórios intermediários requeridos no nome do arquivo se necessário.

Também Veja

Recursos, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2024, MetaQuotes Ltd.

SetReturnError

Define o código de retorno que devolverá um processo do terminal ao encerrar o trabalho.

```
void  SetReturnError(
   int ret_code      // código de encerramento do terminal do cliente
   );
```

Parâmetros

ret_code

[in] O código de retorno que será devolvido pelo processo do terminal do cliente ao encerrar o trabalho.

Valor retornado

Não há nenhum valor de retorno.

Observação

A instalação desse código de retorno *ret_code* usando a função SetReturnError() será útil para analisar os motivos de encerramento do programa ao iniciar o terminal a partir da linha de comando.

Ao contrário da função SetReturnError(), a função TerminalClose() não encerra o terminal, ela apenas define o código de retorno que devolverá o processo do terminal após seu encerramento.

Se a função SetReturnError() foi chamada repetidamente e/eu a partir de diferentes programas MQL5, o terminal retornará o último código de retorno definido.

O código exibido será devolvido no final do processo do terminal, exceto nos seguintes casos:

- se surgir um erro crítico em tempo de execução;
- se for chamada a função TerminalClose(int ret_code) que fornece ao terminal um comando para encerrar o trabalho com um código especificado.

Veja também

Execução de programas, Erros de execução, Causas de desinicialização, TerminalClose

© 2000-2024, MetaQuotes Ltd.

SetUserError

Define a variável predefinida _LastError para o valor igual a ERR_USER_ERROR_FIRST + user_error

```
void  SetUserError(
   ushort user_error,   // número do erro
   );
```

Parâmetros

user_error

[in] Número do erro definido pelo usuário.

Valor do Retorno

Sem valor de retorno.

Observação

Depois de um erro ser definido usando a função SetUserError(user_error), GetLastError() retorna um valor igual a ERR_USER_ERROR_FIRST + user_error.

Exemplo:

```
void OnStart()
  {
//--- define o número de erro 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- obtém o último código de erro
   Print("GetLastError = ",GetLastError());
/*
   Result
   GetLastError = 65537
*/
  }
```

© 2000-2024, MetaQuotes Ltd.

Sleep

A função suspende a execução do Expert Advisor corrente ou script por um intervalo especificado.

```
void  Sleep(
   int  milliseconds      // intervalo
   );
```

Parâmetros

milliseconds

[in] Intervalo de atraso em milissegundos.

Valor do Retorno

Sem valor de retorno.

Observação

A função Sleep() não pode ser chamada por indicadores customizados, porque indicadores são executados na thread de interface e não devem ser atrasados. A função foi incorporada para verificar flag interrompido a cada 0.1 segundos.

© 2000-2024, MetaQuotes Ltd.

TerminalClose

A função ordena o terminal a finalizar a operação.

```
bool  TerminalClose(
   int ret_code      // código de fechamento do terminal cliente
   );
```

Parâmetros

ret_code

[in] Código de retorno, retornado pelo processo do terminal cliente na finalização da operação.

Valor do Retorno

A função retorna true para sucesso, caso contrário - false.

Observação

A função TerminalClose() não interrompe o terminal imediatamente, ela apenas ordena o terminal a finalizar sua operação.

O código de um Expert Advisor que chamou TerminalClose() deve ter todos os arranjos para a finalização imediata (por exemplo, todos os arquivo previamente abertos devem ser fechados no modo normal. A chamada desta função deve ser seguida pelo operador return.

O parâmetro *ret_code* permite indicar o código de retorno necessário para analise dos motivos da finalização de programa da operação do terminal ao inicializá-lo a partir da linha de comando.

Exemplo:

```
//--- parâmetros de entrada
input int  tiks_before=500; // número de ticks até finalização
input int  pips_to_go=15;   // distancia em pips
input int  seconds_st=50;   // número de segundos dados para o Expert Advisor
//--- globais
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," código do motivo = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlTick       tick;
   double        distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid =",first_bid);
      return;
     }
//--- distância do preço em pips
   distance=(tick.bid-first_bid)/_Point;
//--- exibe uma notificação para rastrear a operação do EA
   string comm="A partir do momento da inicial:\r\n\x25CF segundos decorridos: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks recebidos: "+(string)tick_counter+" ;"+
               "\r\n\x25CF preço passou em pontos: "+StringFormat("%G",distance);
   Comment(comm);
//--- seção para verificar condição para fechar o terminal
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // saída pela contador de tick
   if(distance>pips_to_go)
      TerminalClose(1);    // subida pelo número de pips igual ao pips_to_go
   if(distance<-pips_to_go)
      TerminalClose(-1);   // descida pelo número de pips igual ao pips_to_go
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // finalização por timeout
//---
  }
```

Também Veja

Execução de programa, Erros de execução, Motivos para desinicialização

© 2000-2024, MetaQuotes Ltd.

TesterHideIndicators

Define o modo mostrar/ocultar indicadores usados no Expert Advisor. Esta função é usada para controlar a visibilidade dos indicadores utilizados somente durante o teste.

```
void  TesterHideIndicators(
   bool      hide     // sinalizador
   );
```

Parâmetros

hide

[in] Sinalizador para ocultar indicadores durante o teste. Especifique true, se você quiser ocultar os indicadores criados, caso contrário, false.

Valor de retorno

Não.

Observação

Por padrão, no gráfico de teste visual, são apresentados todos os indicadores que são criados no Expert Advisor testado. Além disso, esses indicadores são mostrados no gráfico que se abre automaticamente no final do teste. A função TesterHideIndicators() permite que o desenvolvedor especifique no código do EA a proibição da exibição dos indicadores utilizados.

Para proibir a exibição do indicador usado durante o teste do EA, é necessário, antes de criar seu identificador, chamar a função TesterHideIndicators() com o parâmetro false, dessa maneira todos os indicadores criados após isso serão marcados com o sinalizador de ocultar. Indicadores marcados com o sinalizador de ocultar não são exibidos nem durante o teste visual nem no gráfico que se abre automaticamente após a conclusão do teste.

Para novamente desativar o modo de ocultar os indicadores criados, basta chamar novamente a função TesterHideIndicators(), mas com o parâmetro true. No gráfico de teste, podem ser vistos apenas os indicadores que são criados diretamente a partir do EA testado. Esta regra se aplica apenas aos casos em que não há um modelo na pasta <catálogo_de_dados>MQL5\Profiles\Templates.

Se, na pasta <catálogo_de_dados>MQL5\Profiles\Templates, estiver o modelo especial <nome_do_expert>.tpl, serão exibidos apenas os indicadores desse modelo, durante o teste visual e no gráfico de teste. Neste caso, não serão exibidos os indicadores utilizados no EA testado. Mesmo se, no código do EA, fosse chamada a função TesterHideIndicators() com parâmetro true.

Se, na pasta <catálogo_de_dados>MQL5\Profiles\Templates, não se encontrar o modelo especial <nome_do_expert>.tpl, mas estiver o modelo tester.tpl, então, durante o teste visual e no gráfico de teste, serão exibidos os indicadores do modelo tester.tpl e os indicadores não exibidos devido ao uso da função TesterHideIndicators(). Se não estiver o modelo tester.tpl, em vez dele serão usados os indicadores do modelo default.tpl.

Se o testador de estratégias não encontrar nenhum modelo adequado (<nome_do_expert>.tpl, tester.tpl ou default.tpl), a função TesterHideIndicators() controlará completamente a exibição dos indicadores usados no EA.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
//--- create MACD indicator
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
         printf("Error creating MACD indicator");
         return(false);
        }
   TesterHideIndicators(false);
//--- create EMA indicator and add it to collection
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_
        {
         printf("Error creating EMA indicator");
         return(false);
        }
//--- succeed
   return(true);
  }
```

Veja também

IndicatorRelease

© 2000-2024, MetaQuotes Ltd.

TesterStatistics

A função retorna o valor do parâmetro estatístico especificado calculado baseado em resultados de teste.

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // ID
   );
```

Parâmetros

statistic_id

[in] O ID do parâmetro estatístico a partir da enumeração ENUM_STATISTICS.

Valor do Retorno

O valor do parâmetro estatístico proveniente de resultados de teste.

Observação

A função pode ser chamada de dentro de OnTester() ou OnDeinit() no Provador de Estratégias. Nos outros casos o resultado é indefinido.

© 2000-2024, MetaQuotes Ltd.

TesterStop

Dá o comando para encerrar o programa durante os testes.

```
void  TesterStop();
```

Valor retornado

Não há nenhum valor de retorno.

Observação

A função TesterStop() se destina ao desligamento antecipado geral do EA no agente de teste, por exemplo, ao atingir um número especificado de transações desfavoráveis ou um nível de rebaixamento especificado.

A chamada da TesterStop() é considerada um encerramento normal do teste e, portanto, será chamada a função OnTester() com feedback para o testador de estratégias sobre todas as estatísticas de negociação acumuladas e valores dos critérios de otimização.

A chamada da ExpertRemove() no testador de estratégia também significa o encerramento normal do teste e permite obter estatísticas de negociação, mas, enquanto isso, o EA é descarregado da memória do agente. Nesse caso, para concluir a passagem no próximo conjunto de parâmetros, será necessário mais tempo a fim de recarregar o programa. Portanto, para encerramento antecipado geral do teste, o uso da TesterStop() é a opção preferida.

Veja também

Execução de programas, Teste de estratégias de negociação, ExpertRemove, SetReturnError

© 2000-2024, MetaQuotes Ltd.

TesterDeposit

Função especial para emular operações de depósito durante o teste. Pode ser usado em alguns sistemas de gerenciamento de capital.

```
bool  TesterDeposit(
   double money      // tamanho do montante a ser depositado
   );
```

Parâmetros

money

[in] Quantidade de dinheiro a ser depositado na moeda de depósito.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Veja também

TesterWithdrawal

© 2000-2024, MetaQuotes Ltd.

TesterWithdrawal

Função especial para emular a operação de saque de dinheiro no processo de teste. Pode ser usado em alguns sistemas de gerenciamento de ativos.

```
bool  TesterWithdrawal(
   double money      // a soma para sacar
   );
```

Parâmetros

money

[in] A soma de dinheiro que nós precisamos sacar (no moeda de depósito).

Valor do Retorno

Com sucesso, retorna verdadeiro (true), do contrário - falso.

Veja também

TesterDeposit

© 2000-2024, MetaQuotes Ltd.

TranslateKey

Retorna o caractere Unicode de acordo com o código de tecla virtual, tendo em conta o idioma de entrada atual e o status das teclas de controle.

```
short  TranslateKey(
   int  key_code      // código de teclas para obter o caractere Unicode
   );
```

Parâmetros

key_code

[in] Código de tecla.

Valor de retorno

Caractere Unicode no caso de uma conversão bem-sucedida. No caso de erro, a função retorna -1.

Observação

A função utiliza ToUnicodeEx para converter as teclas digitadas - pelo usuário - para caracteres Unicode. O erro pode surgir se não funcionar ToUnicodeEx, por exemplo, ao tentar obter o caractere para a tecla SHIFT.

Exemplo:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- se o caractere digitado for convertido com sucesso para Unicode
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Error in TranslateKey for key=",lparam);
     }
  }
```

Veja também

Eventos do terminal de cliente, OnChartEvent

© 2000-2024, MetaQuotes Ltd.

ZeroMemory

A função redefine uma variável passada para ela por referência.

```
void  ZeroMemory(
   void & variable      // redefine variável
   );
```

Parâmetros

variable

[in] [out] Variável passada por referência, que você quer redefinir (inicializar com zeros).

Valor do Retorno

Sem valor de retorno.

Observação

Se o parâmetro da função é uma string, a chamada será equivalente a indicar NULL como seu valor.
Para tipos simples e seus arrays, bem como para estruturas/classes consistindo de tais tipos, isso é uma simples redefinição.
Para objetos contendo strings e arrays dinâmicos, ZeroMemory() é chamada para cada elemento.
Para quaisquer arrays não protegidos pelo modificador const, todos os elementos são zerados.
Para arrays de objetos complexos, ZeroMemory() é chamada para cada elemento.

ZeroMemory() não pode ser aplicada a classes com membros protegidos ou herança.

© 2000-2024, MetaQuotes Ltd.

Grupo de Funções para Trabalhar com Arrays

Arrays podem ter no máximo quatro dimensões. Cada dimensão é indexada de 0 a *dimension_size-1*. No caso particular de um array de uma dimensão de 50 elementos, a chamada do primeiro elemento aparece como array[0], do último elemento - como array[49].

Função	Ação
ArrayBsearch	Retorna o índice do primeiro elemento encontrado na primeira dimensão do array
ArrayCopy	Copia um array em outro array
ArrayCompare	Retorna o resultado da comparação entre dois arrays de tipos simples ou estruturas personalizados sem objetos complexos
ArrayFree	Libera buffer de qualquer array dinâmico e define o tamanho da dimensão zero em 0.
ArrayGetAsSeries	Verifica a direção da indexação de um array
ArrayInitialize	Define todos os elementos de um array numérico para um único valor
ArrayFill	Preenche um array com o valor especificado
ArrayIsSeries	Verifica se um é array é uma série de tempo
ArrayIsDynamic	Verifica se um array é dinâmico
ArrayMaximum	Busca por um elemento com o valor máximo
ArrayPrint	No log, exibe uma matriz de tipo ou estrutura simples
ArrayMinimum	Busca por um elemento com o valor mínimo
ArrayRange	Retorna o número de elementos na dimensão especifica do array
ArrayResize	Define o novo tamanho da primeira dimensão do array
ArrayInsert	Insere na matriz de destino o número especificado de elementos da matriz de origem, iniciando no índice especificado
ArrayRemove	Remove o número especificado de elementos da matriz, iniciando no índice especificado
ArrayReverse	Expande o número de elementos especificado na matriz, iniciando no índice especificado
ArraySetAsSeries	Define a direção de indexação de um array
ArraySize	Retorna o número de elemento no array
ArraySort	Ordena arrays números pela primeira dimensão
ArraySwap	Troca entre si o conteúdo de duas matrizes do tipo dinâmico

© 2000-2024, MetaQuotes Ltd.

ArrayBsearch

Procura um valor especificado em um array numérico multidimensional ordenado em ordem crescente. A pesquisa é realizada através dos elementos da primeira dimensão.

Para buscar em um array de tipo double

```
int  ArrayBsearch(
   const double&    array[],   // array para busca
   double           value      // o que é pesquisada por
   );
```

Para buscar em um array de tipo float

```
int  ArrayBsearch(
   const float&     array[],   // array para busca
   float            value      // o que é pesquisada por
   );
```

Para buscar em um array de tipo long

```
int  ArrayBsearch(
   const long&      array[],   // array para busca
   long             value      // o que é pesquisada por
   );
```

Para buscar em um array de tipo int

```
int  ArrayBsearch(
   const int&       array[],   // array para busca
   int              value      // o que é pesquisada por
   );
```

Para buscar em um array de tipo short

```
int  ArrayBsearch(
   const short&     array[],   // array para busca
   short            value      // o que é pesquisada por
   );
```

Para buscar em um array de tipo char

```
int  ArrayBsearch(
   const char&      array[],   // array para busca
   char             value      // o que é pesquisada por
   );
```

Parâmetros

array[]

[in] Array numérico para busca.

value

© 2000-2024, MetaQuotes Ltd.

[in] Valor para busca.

Valor do Retorno

A função retorna o índice de um elemento encontrado. Se o valor desejado não for encontrado, a função retorna o índice de um elemento mais próximo em valor.

Observação

A busca binária processa somente arrays ordenados. Para ordenar arrays numéricos use a função ArraySort().

Exemplo:

```
#property description "Script baseado no indicador RSI para visualização de dados"
#property description "quantas vezes o mercado ficou em"
#property description "áreas sobre-compradas e sobre-vendidas em intervalo de tempo es
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input int                 InpMAPeriod=14;                    // Período de média móvel
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;       // Tipo de preço
input double              InpOversoldValue=30.0;             // Nível sobre-vendido
input double              InpOverboughtValue=70.0;           // Nível sobre-comprado
input datetime            InpDateStart=D'2012.01.01 00:00';  // Data de início para ana
input datetime            InpDateFinish=D'2013.01.01 00:00'; // Data final para análise
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // array do valores do indicador
   int    size=0;     // tamanho de array
//--- receber a manipulação do indicador RSI
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- falha para receber a manipulação do indicador
      PrintFormat("Manipulador de indicator recebido com erro. Código de erro = %d",Ge
      return;
     }
//--- fica no loop, até que o indicador calcula todos os seus valores
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- sair se o indicador concluir forçadamente a operação de script
      if(IsStopped())
         return;
      //--- uma pausa para permitir que o indicador calcule todos os seus valores
      Sleep(10);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- copiar os valores do indicador para um determinado período de tempo
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Falha para copiar valores do indicador. Código de erro = %d",GetLas
      return;
     }
//--- recebe o tamanho do array
   size=ArraySize(rsi_buff);
//--- classificar o array
   ArraySort(rsi_buff);
//--- descobrir o tempo (em termos percentuais) que o mercado ficou na área sobre-vend
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- descobrir o tempo (em termos percentuais) que o mercado ficou na área sobre-comp
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- formar as strings para exibir os dados
   string str="A partir de "+TimeToString(InpDateStart,TIME_DATE)+" para "
              +TimeToString(InpDateFinish,TIME_DATE)+" o mercado ficou:";
   string str_ovb="em área sobre-comprada "+DoubleToString(ovb,2)+"% do tempo";
   string str_ovs="em área sobre-vendida "+DoubleToString(ovs,2)+"% do tempo";
//--- mostrar os dados no gráfico
   CreateLabel("topo",5,60,str,clrDodgerBlue);
   CreateLabel("sobre-comprado",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("sobre-vendido",5,10,str_ovs,clrDodgerBlue);
//--- redesenhar o gráfico
   ChartRedraw(0);
//--- pausa
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Mostrar comentário no canto inferior esquerdo do gráfico         |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- cria um rótulo (label)
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- colocar a etiqueta no canto inferior esquerdo
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- alterar a posição do ponto de ancoragem
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- distância a partir do ponto de ancoragem na direção X
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- distância a partir do ponto de ancoragem na direção Y
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- texto do rótulo
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- cor do texto
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tamanho do texto
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayCopy

Copia um array em um outro array.

```
int  ArrayCopy(
   void&        dst_array[],         // array de destino
   const void&  src_array[],         // array de origem
   int          dst_start=0,         // índice de início do array destino a partir do
   int          src_start=0,         // primeiro índice de um array de origem
   int          count=WHOLE_ARRAY    // número de elementos
   );
```

Parâmetros

dst_array[]

[out] Array destino

src_array[]

[in] Array fonte

dst_start=0

[in] Índice de início do array destino. Por padrão, o índice de início é 0.

src_start=0

[in] Índice de início para o array fonte. Por padrão, o índice de início é 0.

count=WHOLE_ARRAY

[in] Número de elementos que devem ser copiados. Por padrão, todo conjunto array é copiado (count=WHOLE_ARRAY).

Valor do Retorno

Retorna o número de elementos copiados.

Observação

Se count<0 ou count>src_size-src_start, toda a parte restante do array é copiada. Arrays são copiados da esquerda para direita. Para arrays de série, a posição de início é corretamente ajustada para copiar da esquerda para direita.

Se os arrays são de tipos diferentes, durante a cópia a função tenta transformar cada elemento do array fonte para o tipo do array destino. Um array de string pode ser copiado somente em um array de string. Arrays de classes e estruturas contendo objetos que requerem inicialização não são copiados. Um array de estruturas pode ser copiado somente em um array do mesmo tipo.

For dynamic arrays with indexing as in timeseries, the size of a destination array is automatically increased to the amount of copied data (if the latter exceeds the array size). The destination array size is not decreased automatically.

Exemplo:

```
#property description "O indicador destaca os candlesticks que são local"
#property description "altos e baixos. Comprimento do intervalo para encontrar"
#property description "valores extremos usando parâmetros de entrada."
```

© 2000-2024, MetaQuotes Ltd.

```
//--- configurações do indicador
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plotar
#property indicator_label1  "Extremums"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- constante pré-definida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parâmetros de entrada
input int InpNum=4; // Meio intervalo de comprimento
//--- buffers do indicador
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- variáveis globais
int    ExtStart=0; // índice do primeiro candlestick que não é um extremo
int    ExtCount=0; // número de não extremos no intervalo
//+------------------------------------------------------------------+
//| Preenchimento dos candlesticks não extremos                      |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- preencher os candlesticks
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- especificar o valor que não é apresentado
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- especificar os nomes dos buffers do indicador para exibir na janela de dados
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- definir a indexação direta em séries de tempo
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- variável da barra para iniciar cálculo
   int start=prev_calculated;
//--- cálculo não é realizado pela primeira InpNum*2 barras
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- se a barra acabou de se formar, verifique o seguinte extremo potencial
   if(rates_total-start==1)
      start--;
//--- índice da barra a ser verificado para o extremo
   int ext;
//--- valor do indicador para cálculo do loop
   for(int i=start;i<rates_total-1;i++)
     {
      //--- inicialmente num bar sem desenhar
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- índice extremo para verificação
      ext=i-InpNum;
      //--- verificar o máximo local
      if(IsMax(high,ext))
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- destacar um candlestick extremo
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- destacar outros candles até o extremo com uma cor neutra
         FillCandles(open,high,low,close);
         //--- alterar as cores variáveis
         ExtStart=ext+1;
         ExtCount=0;
         //--- passar para a próxima iteração
         continue;
        }
      //--- verifique o mínimo local
      if(IsMin(low,ext))
        {
         //--- destacar um candlestick extremo
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- destacar outros candles até o extremo com uma cor neutra
         FillCandles(open,high,low,close);
         //--- alterar os valores das variáveis
         ExtStart=ext+1;
         ExtCount=0;
         //--- passar para a próxima iteração
         continue;
        }
      //--- aumentar o número de não extremos no intervalo
      ExtCount++;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Verificar se o atual elemento array é um local máximo (high)     |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- variável do início do intervalo
   int i=ind-InpNum;
//--- final do período do intervalo
   int finish=ind+InpNum+1;
//--- verificar a primeiro metade do intervalo
   for(;i<ind;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       if(price[ind]<=price[i])
          return(false);
      }
//--- verificar a segunda metade do intervalo
   for(i=ind+1;i<finish;i++)
      {
       if(price[ind]<=price[i])
          return(false);
      }
//--- este é um extremo
   return(true);
  }
//+------------------------------------------------------------------+
//| Verificar se o atual elemento array é um local mínimo (low)      |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- variável do início do intervalo
   int i=ind-InpNum;
//--- variável do final do intervalo
   int finish=ind+InpNum+1;
//--- verificar a primeiro metade do intervalo
   for(;i<ind;i++)
      {
       if(price[ind]>=price[i])
          return(false);
      }
//--- verificar a segunda metade do intervalo
   for(i=ind+1;i<finish;i++)
      {
       if(price[ind]>=price[i])
          return(false);
      }
//--- este é um extremo
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayCompare

A função retorna o resultado da comparação de dois arrays de mesmo tipo. Pode ser usado para comparar arrays de tipos simples ou estruturas customizadas sem objetos complexos, isto é, estruturas customizadas que não contém strings, arrays dinâmicos, classes e outras estruturas com objetos complexos.

```
int  ArrayCompare(
   const void&  array1[],            // primeiro array
   const void&  array2[],            // segundo array
   int          start1=0,            // deslocamento inicial no primeiro array
   int          start2=0,            // deslocamento inicial no segundo array
   int          count=WHOLE_ARRAY    // número de elementos para comparação
   );
```

Parâmetros

array1[]

[in] Primeiro array.

array2[]

[in] Segundo array.

start1=0

[in] O índice inicial do elemento no primeiro array, a partir do qual a comparação começa. O índice de começo default - 0.

start2=0

[in] O índice inicial do elemento no segundo array, a partir do qual a comparação começa. O índice de começo default - 0.

count=WHOLE_ARRAY

[in] Número de elementos a ser comparados. Todos os elementos de ambos os arrays participam da comparação por default (count=WHOLE_ARRAY).

Valor de retorno

- -1, se array1[] menos que array2[]
- 0, se array1[] igual a array2[]
- 1, se array1[] mais que array2[]
- -2, se um erro ocorrer devido a incompatibilidade dos tipos dos arrays comparados ou se os valores start1, start2 ou count levarem a cair para fora do array.

Observação

A função não retornará 0 (os arrays não serão considerados iguais) se os arrays diferirem em tamanho e count=WHOLE_ARRAY para o caso em que um array é um subconjunto fiel do outro. Neste caso, o resultado da comparação de tamanhos destes arrays irão retornar: -1, se o tamanho de array1[] for menor que o tamanho de array2[] , caso contrário 1.

© 2000-2024, MetaQuotes Ltd.

ArrayFree

Libera um buffer de qualquer array dinâmico e define o tamanho da dimensão zero para 0.

```
void  ArrayFree(
   void&  array[]      // array
   );
```

Parâmetros

array[]

[in] Array dinâmico.

Valor do Retorno

Sem valor de retorno.

Observação

A necessidade de usar a função ArrayFree() pode não aparecer com muita freqüência, considerando que toda a memória utilizada é liberada de uma só vez e o principal trabalho com os arrays compreende o acesso aos buffers dos indicadores. Os tamanhos dos buffers são controlados automaticamente pelo subsistema executivo do terminal.

No caso, é necessário controlar manualmente a memória no ambiente dinâmico complexo da aplicação, a função ArrayFree() permite que os usuários possam liberar a memória ocupada pelo array dinâmico desnecessário de forma direta e imediata.

Exemplo:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- constantes predefinidas
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| Uma classe para trabalhar com uma memória                        |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- tamanho de array
   int               m_arr_size;
   //--- arrays
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
   double            m_arr_double[];
```

© 2000-2024, MetaQuotes Ltd.

```
   long              m_arr_long[];
   //--- rótulos
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- botões
   CButton           m_button_add;
   CButton           m_button_free;
   //--- caixas de combinação
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- valor atual do tipo array da caixa de combinação
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- método de criação de objeto de classe
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- manipulador de eventos gráfico
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- cria um rótulos
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- criar elementos de controle
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- manipuladores de evento
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- métodos para trabalhar com o atual array
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| Liberar memória do atual array                                   |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
//--- restabelecer o tamanho do array
```

© 2000-2024, MetaQuotes Ltd.

```
   m_arr_size=0;
//--- liberar o array
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| Tentativa de adicionar memória para o atual array                |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- se o tamanho da memória usada excede o tamanho da memória física, sair
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOI
      return(false);
//--- tentativa para alocar memória de acordo com o atual tipo
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- memória alocada
   return(true);
  }
//+------------------------------------------------------------------+
//| Manipulação de eventos                                           |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Construtor                                                       |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Destrutor                                                        |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| Método para criação de objeto de classe                          |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- base criada de objetos de classe
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- preparar strings para etiquetas
   string str_physical="Memória Física = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   string str_total="Memória total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOT
   string str_available="Memória disponível = "+(string)TerminalInfoInteger(TERMINAL_M
   string str_used="Memória usada = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED
//--- cria um rótulos
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type =
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- create control elements
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Adicionar",12,cl
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Livre",12,clrR
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
//--- inicializar a variável
```

© 2000-2024, MetaQuotes Ltd.

```
   m_arr_size=0;
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Criar o botão                                                    |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- criar o botão
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- texto
   if(!button.Text(str))
      return(false);
//--- tamanho de fonte
   if(!button.FontSize(font_size))
      return(false);
//--- cor de rótulo
   if(!button.Color(clr))
      return(false);
//--- adicionar o botão para os elementos de controle
   if(!Add(button))
      return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Criar uma caixa de combinação para o tamanho do array            |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- criar a caixa de combinação
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- adicionar elementos para a caixa de combinação
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- definir o elemento atual da caixa de combinação
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- adicionar a caixa de combinação para controlar elementos
   if(!Add(m_combo_box_step))
      return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Criar uma caixa de combinação para o tipo de array               |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- criar a caixa de combinação
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_STARI
      return(false);
//--- adicionar elementos para a caixa de combinação
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- definir o elemento atual da caixa de combinação
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- armazenar o elemento da caixa de combinação atual
   m_combo_box_type_value=3;
//--- adicionar a caixa de combinação para controlar elementos
   if(!Add(m_combo_box_type))
      return(false);
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| criar um rótulo                                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- criar um rótulo
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- texto
   if(!lbl.Text(str))
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- tamanho de fonte
   if(!lbl.FontSize(font_size))
      return(false);
//--- cor
   if(!lbl.Color(clr))
      return(false);
//--- adicionar a etiqueta para controlar elementos
   if(!Add(lbl))
      return(false);
//--- sucesso
   return(true);
  }
//+------------------------------------------------------------------+
//| Manipulador criando botão de evento "Adicionar"                  |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- aumentar o tamanho do array
   m_arr_size+=(int)m_combo_box_step.Value();
//--- tentativa de alocar memória para o atual array
   if(CurrentArrayAdd())
     {
      //--- memória alocada, exibir o status atual na tela
      m_lbl_memory_available.Text("Memória disponível = "+(string)TerminalInfoInteger
      m_lbl_memory_used.Text("Memória usada = "+(string)TerminalInfoInteger(TERMINAL_M
      m_lbl_array_size.Text("Tamanho de array = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- falha para alocar memória, exibir a mensagem de erro
      m_lbl_error.Text("Array é muito grande, erro!");
      //--- retornar o tamanho do array anterior
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| Manipulador criando o botão de evento "Livre"                    |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- liberar memória do atual array
   CurrentArrayFree();
//--- exibir o status atual na tela
   m_lbl_memory_available.Text("Memória disponível = "+(string)TerminalInfoInteger(TER
   m_lbl_memory_used.Text("Memória usada = "+(string)TerminalInfoInteger(TERMINAL_MEMO
   m_lbl_array_size.Text("Tamanho de array = 0");
   m_lbl_error.Text("");
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Manipulador da caixa de combinação para alterar evento           |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- verificar se o tipo de array foi alterado
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- liberar memória do atual array
      OnClickButtonFree();
      //--- trabalhar com outro tipo de array
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- exibir um novo tipo de array na tela
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Tipo de array = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Tipo de array = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Tipo de array = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Tipo de array = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Tipo de array = long");
     }
  }
//--- objeto de classe CMemoryControl
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- criar o diálogo
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- lançamento
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayGetAsSeries

Verifica a direção do índice de um array.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // array para verificação
   );
```

Parâmetros

array

[in] Array verificado.

Valor do Retorno

Retorna true, se o array especificado tem o flag AS_SERIES definido, isto é, o acesso ao array é realizado de trás para frente como em séries de tempo. A indexação em timeséries de tempo difere da indexação de um array comum, na medida que os elementos de séries de tempo são indexados do fim para o começo (dos dados mais novos para os mais antigos).

Observação

Para verificar se um array se comporta como uma série de tempo, use a função ArrayIsSeries(). Array de dados de preços passados como parâmetros de entrada na função OnCalculate() não obrigatoriamente têm a mesma direção de uma série de tempo. A direção de indexação necessária pode ser definida usando a função ArraySetAsSeries().

Exemplo:

```
#property description "Indicador calcula os valores absolutos das diferenças entre"
#property description "Preços de Open (Abertura) e Close (Fechamento) ou High (Alto) e
#property description "como um histograma."
//--- configurações do indicador
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- parâmetros de entrada
input bool InpAsSeries=true; // Direção de indexação no buffer de indicador
input bool InpPrices=true;   // Cálculo de preços (true - Abertura, Fechamento; false
//--- buffer do indicador
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Calcule os valores do indicador                                  |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- iniciar variável para o cálculo das barras
```

© 2000-2024, MetaQuotes Ltd.

```
   int start=prev_calculated;
//--- trabalho na última barra se os valores do indicador já foram calculados no tick
   if(prev_calculated>0)
      start--;
//--- definir a direção de indexação nos arrays
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- substituir a direção de indexação com um direto, se necessário
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- calcule valores do indicador
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- vincular buffers de indicador
   SetIndexBuffer(0,ExtBuffer);
//--- conjunto de elementos de indexação no buffer de indicador
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- verificar quais os preços do indicador são calculados
   if(InpPrices)
     {
      //--- Preços Open (abrir) e Close (fechar)
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- definir a cor do indicador
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- Preços High (máximo) e Low (mínimo)
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- definir a cor do indicador
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- calcular o indicador de acordo com o valor do sinalizador
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Aceso a séries de tempo, ArraySetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArrayInitialize

A função inicializa um array numérico com um valor predefinido.

For initialization of an array of char type

```
int  ArrayInitialize(
   char    array[],     // array inicializado
   char    value        // valor que será definido
   );
```

For initialization of an array of short type

```
int  ArrayInitialize(
   short   array[],     // array inicializado
   short   value        // valor que será definido
   );
```

For initialization of an array of int type

```
int  ArrayInitialize(
   int     array[],     // array inicializado
   int     value        // valor que será definido
   );
```

For initialization of an array of long type

```
int  ArrayInitialize(
   long    array[],     // array inicializado
   long    value        // valor que será definido
   );
```

For initialization of an array of float type

```
int  ArrayInitialize(
   float   array[],     // array inicializado
   float   value        // valor que será definido
   );
```

For initialization of an array of double type

```
int  ArrayInitialize(
   double  array[],     // array inicializado
   double  value        // valor que será definido
   );
```

For initialization of an array of bool type

```
int  ArrayInitialize(
   bool    array[],     // array inicializado
   bool    value        // valor que será definido
   );
```

© 2000-2024, MetaQuotes Ltd.

For initialization of an array of uint type

```
int  ArrayInitialize(
   uint    array[],      // array inicializado
   uint    value         // valor que será definido
   );
```

Parâmetros

array[]

[out] Array numérico que deve ser inicializado.

value

[in] Novo valor que deve ser definido para todos os elementos do array.

Valor do Retorno

Número de elementos.

Observação

A função ArrayResize() permite definir o tamanho de um array com uma reserva para expansão posterior sem a realocação de memória. É implementada para uma melhor performance, porque as operações de realocação de memória são razoavelmente lentas.

A inicialização de um array usando ArrayInitialize(array, init_val) não significa a inicialização com o mesmo valor dos elementos da reserva alocados para este array. Em uma expansão posterior do *array* usando a função ArrayResize(), os elementos serão adicionados no final do array, seus valores serão indefinidos e na maioria dos casos não serão iguais ao *init_value*.

Exemplo:

```
void OnStart()
  {
//--- array dinâmico
   double array[];
//--- vamos definir o tamanho do array para 100 elementos e reservar um buffer para ou
   ArrayResize(array,100,10);
//--- inicializa os elementos do array com valor EMPTY_VALUE=DBL_MAX
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Valores dos 10 últimos elementos antes da inicialização");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- expande o array em 5 elementos
   ArrayResize(array,105);
   Print("Valores dos 10 últimos elementos antes do ArrayResize(array,105)");
//--- valores dos 5 últimos elementos são obtidos do buffer de reserva
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayFill

A função preenche um array com o valor especificado.

```
void  ArrayFill(
   void&  array[],       // array
   int    start,         // índice de início
   int    count,         // número de elementos para preencher
   void   value          // valor
   );
```

Parâmetros

array[]

[out] Array de tipo simples (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Índice de início. Neste caso, flag AS_SERIES específico é ignorado.

count

[in] Número de elementos para preencher.

value

[in] Valor usado no preenchimento do array.

Valor de retorno

Sem valor de retorno.

Observação

Quando a função ArrayFill() é chamada, a direção de indexação normal(da esquerda para direita) é sempre implícita. Isso significa que a alteração da ordem de acesso dos elementos do array usando a função ArraySetAsSeries() é ignorada.

Um array multidimensional é exibido como unidimensional quando processado pela função ArrayFill(). Por exemplo, array[2][4] é processado como array[8]. Portanto, você deve especificar o índice do elemento inicial para ser igual a 5 quando trabalhar com este array. Assim, a chamada de ArrayFill(array, 5, 2, 3.14) para array[2][4] preenche os elementos array[1][1] e array[1][2] com 3.14.

Exemplo:

```
void OnStart()
  {
//--- declara um array dinâmico
   int a[];
//--- define tamanho
   ArrayResize(a,10);
//--- preenche os primeiros 5 elementos com 123
   ArrayFill(a,0,5,123);
//--- preenche os primeiros 5 elementos com 456
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayFill(a,5,5,456);
//--- mostrar valores
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayIsDynamic

A função verifica se um array é dinâmico.

```
bool  ArrayIsDynamic(
   const void&  array[]    // array verificado
   );
```

Parâmetros

array[]

[in] Array verificado.

Valor do Retorno

Retorna true se o array selecionado é dinâmico, caso contrário retorna false.

Exemplo:

```
#property description "Este indicador não calcula valores. Faz uma única tentativa pa
#property description "aplicar a chamada da função ArrayFree() para três arrays: dinâ
#property description "um buffer de indicador. Os resultados são mostrados no jornal 
//--- configurações do indicador
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- variáveis globais
double ExtDynamic[];   // array dinâmico
double ExtStatic[100]; // array estático
bool   ExtFlag=true;   // bandeira
double ExtBuff[];      // buffer do indicador
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- alocar memória para o array
   ArrayResize(ExtDynamic,100);
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- executar uma análise única
   if(ExtFlag)
     {
      //--- tentativa para liberar memória para os arrays
      //--- 1. Array dinâmico
      Print("+============================+");
      Print("1. Verificar o array dinâmico:");
      Print("Antes do tamanho da memória ser liberada = ",ArraySize(ExtDynamic));
      Print("Isto é um array dinâmico = ",ArrayIsDynamic(ExtDynamic) ? "Sim" : "Não");
      //--- tentativa de liberar memória do array
      ArrayFree(ExtDynamic);
      Print("Após o tamanho da memória ser liberada = ",ArraySize(ExtDynamic));
      //--- 2. Array estático
      Print("2. Verificar array estático:");
      Print("Antes do tamanho da memória ser liberada = ",ArraySize(ExtStatic));
      Print("Isto é um array dinâmico = ",ArrayIsDynamic(ExtStatic) ? "Sim" : "Não");
      //--- tentativa de liberar memória do array
      ArrayFree(ExtStatic);
      Print("Após do tamanho da memória ser liberada = ",ArraySize(ExtStatic));
      //--- 3. Buffer do indicador
      Print("3. Verificar buffer do indicador:");
      Print("Antes do tamanho da memória ser liberada = ",ArraySize(ExtBuff));
      Print("Isto é um array dinâmico = ",ArrayIsDynamic(ExtBuff) ? "Sim" : "Não");
      //--- tentativa de liberar memória do array
      ArrayFree(ExtBuff);
      Print("Após do tamanho da memória ser liberada = ",ArraySize(ExtBuff));
      //--- alterar o valor do sinalizador
      ExtFlag=false;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Acesso a séries de tempo e indicadores

© 2000-2024, MetaQuotes Ltd.

ArrayIsSeries

A função verifica se um array é uma série de tempo.

```
bool  ArrayIsSeries(
   const void&  array[]    // array verificado
   );
```

Parâmetros

array[]

[in] Array verificado.

Valor do Retorno

Retorna true, se o array verificado é um array de série de tempo, caso contrário, retorna false. Arrays passados como parâmetro para a função OnCalculate() devem ser verificados quanto a ordem de acesso dos elementos do array através de ArrayGetAsSeries().

Exemplo:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Etiqueta1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers do indicador
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[] é timeseries");
   else
      Print("open[] não é timeseries!!!");
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Acesso a séries de tempo e indicadores

© 2000-2024, MetaQuotes Ltd.

ArrayMaximum

Procura o maior elemento na primeira dimensão de um array numérico multidimensional.

```
int  ArrayMaximum(
   const void&   array[],            // array para busca
   int           start=0,            // índice a partir do qual começa a verificação
   int           count=WHOLE_ARRAY   // número de elementos verificados
   );
```

Parâmetros

array[]

[in] Um array númerico, na qual a busca é feita.

start=0

[in] Índice a partir do qual começa a verificação.

count=WHOLE_ARRAY

[in] Número de elementos para busca. Por padrão, busca no array inteiro (count=WHOLE_ARRAY)..

Valor do Retorno

A função retorna um índice de um elemento encontrado levando em consideração a série do array. Em caso de falha, retorna -1.

Observação

O valor da bandeira AS_SERIES É levado em consideração durante a procura de um máximo.

Funções ArrayMaximum e ArrayMinimum aceitam arrays multi-dimensionais como um parâmetro. No entanto, a pesquisa é sempre aplicada à primeira (zero) dimensão.

Exemplo:

```
#property description "O indicador mostra sobre as barras (candlesticks) um prazo de t
//--- configurações do indicador
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2024, MetaQuotes Ltd.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constante pré-definida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parâmetros de entrada
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Prazo de tempo para calcula
input datetime        InpDateStart=D'2013.01.01 00:00'; // Analisar a partir desta dat
//--- buffers do indicador para candlesticks de baixa
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- buffers do indicador para candlesticks de alta
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variáveis globais
datetime ExtTimeBuff[];      // buffer de tempo para maior periodicidade
int      ExtSize=0;          // tamanho do buffer de tempo
int      ExtCount=0;         // índice dentro do buffer de tempo
int      ExtStartPos=0;      // posição inicial para cálculo do indicador
bool     ExtStartFlag=true;  // sinalizador auxiliar para receber a posição inicial
datetime ExtCurrentTime[1];  // último tempo na geração da barra de maior periodicidad
datetime ExtLastTime;        // último tempo a partir da barra de maior periodicidade,
bool     ExtBearFlag=true;   // flag para definir a ordem de escrita do dados para os
bool     ExtBullFlag=true;   // flag para definir a ordem de escrita do dados para os
int      ExtIndexMax=0;      // índice do elemento máximo do array
int      ExtIndexMin=0;      // índice do elemento mínimo do array
int      ExtDirectionFlag=0; // direção de movimento de preços para o atual candlestic
//--- alternar entre o preço de abertura e fechamento do candlestick para desenhar co
```

© 2000-2024, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+-----------------------------------------------------------------+
//| Preenchendo a parte básica do candlestick                       |
//+-----------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- encontrar o índice dos elementos máximo e mínimo nos arrays
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // máximo em Alta
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // mínimo em Baixa
//--- definir quantas barras do período de tempo atual serão preenchidas
   int count=fill_index-start+1;
//--- se o preço de fechamento no primeiro bar é excede ao do último bar, o candlestic
   if(open[start]>close[last])
     {
      //--- antes que o candlestick seja de alta, limpar os valores de alta dos buffer
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- candlestick de baixa
      ExtDirectionFlag=-1;
      //--- gerar o candlestick
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- sair da função
      return;
     }
//--- se o preço de fechamento no primeiro bar é menor do que último bar, o candlestic
   if(open[start]<close[last])
     {
      //--- antes que o candlestick seja de baixa, limpar os valores de baixa dos buf
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- candlesticks de alta
      ExtDirectionFlag=1;
      //--- gerar o candlestick
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- sair da função
      return;
     }
//--- se você está nesta parte da função, o preço de abertura no primeiro bar é igual
//--- preço de fechamento no último bar; então o candlestick é considerado de baixa
//--- antes que o candlestick seja de alta, limpar os valores de alta dos buffers do
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- candlestick de baixa
   ExtDirectionFlag=-1;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se os preços de fechamento e de abertura são iguais, use o deslocamento para a e
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Preencher o final do candlestick                                 |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- não desenhar em caso de uma única barra
   if(last-start==0)
      return;
//--- se o preço de fechamento no primeiro bar é excede ao do último bar, o candlestic
   if(open[start]>close[last])
     {
      //--- gerar o final do candelstick
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- sair da função
      return;
     }
//--- se o preço de fechamento no primeiro bar é menor do que último bar, o candlestic
   if(open[start]<close[last])
     {
      //--- gerar o final do candelstick
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- sair da função
      return;
     }
//--- se você está nesta parte da função, o preço de abertura no primeiro bar é igual
//--- preço de fechamento no último bar; então o candlestick é considerado de baixa
//--- gerar o final do candelstick
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificar o período do indicador
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- exibir dados de preço num primeiro plano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- vincular buffers do indicador
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- definir alguns valores de propriedade para a criação do indicador
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo de construção gráfic
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // estilo de linha de desenh
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // largura de linha de deser
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- neste caso ainda não existe cálculos de barras
   if(prev_calculated==0)
     {
      //--- receber a maior periodicidade das barra no tempo de chegada
      if(!GetTimeData())
         return(0);
     }
//--- definir a indexação direta
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- iniciar variável para o cálculo das barras
   int start=prev_calculated;
//--- se uma barra é gerada, recalcular o valor do indicador sobre ela
   if(start!=0 && start==rates_total)
      start--;
//--- o loop para cálculos dos valores do indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- preencher i elementos dos buffers do indicador por valores vazios
      FillIndicatorBuffers(i);
      //--- executar cálculos para as barras iniciando a partir dos dados InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definir posição, a partir do qual os valores são exibidos pela primeira
         if(ExtStartFlag)
           {
            //--- armazenar o número do bar inicial
            ExtStartPos=i;
            //--- definir a primeira data a partir da maior periodicidade excedendo t
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- alterar o valor da sinalização em ordem não executada neste bloco n
            ExtStartFlag=false;
           }
         //--- verificar se ainda existem quaisquer elementos no array
         if(ExtCount<ExtSize)
           {
            //--- esperar o valor da periodicidade atual para chegar a um maior períod
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- desenhar a parte principal do candlestick (sem preencher o espaço
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- preencher o final do candlestick(a área entre a última e a penúlt
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtI
```

```
                //--- deslocar a posição inicial para desenhar o próximo candlestick
                ExtStartPos=i;
                //--- aumentar o contador de array
                ExtCount++;
               }
            else
                continue;
           }
         else
           {
            //--- redefinir os valores de array
            ResetLastError();
            //--- receber a última data do maior período de tempo
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
              {
               Print("Erro na cópia dos dados, código = ",GetLastError());
               return(0);
              }
            //--- se a nova data for posterior, parar de gerar o candlestick
            if(ExtCurrentTime[0]>ExtLastTime)
              {
               //--- limpar a área entre a última e a penúltima barra nos principais b
               ClearEndOfBodyMain(i-1);
               //--- preencher a área usando buffers auxiliares do indicador
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- deslocar a posição inicial para desenhar o próximo candlestick
               ExtStartPos=i;
               //--- restabelecer flag da direção do preço
               ExtDirectionFlag=0;
               //--- armazenar a nova última data
               ExtLastTime=ExtCurrentTime[0];
              }
            else
              {
               //--- gerar o candlestick
               FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Verificar a exatidão do período do indicador determinado         |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- o período do indicador deve ultrapassar o período de tempo que é exibido sobre
```

```
   if(current_period>=high_period)
     {
      Print("Erro! O valor do período do indicador deve exceder o valor do intervalo c
      return(false);
     }
//--- se o período do indicador é de uma semana ou um mês, o período está correto
   if(high_period>32768)
      return(true);
//--- converter valores do período para minutos
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- o período do indicador deve ser múltiplo do período de tempo, é exibido sobre
   if(high_period%current_period!=0)
     {
      Print("Erro! O valor do período do indicador deve ser múltiplo ao valor do atual
      return(false);
     }
//--- o período do indicador deve exceder o prazo de tempo quando apresentado de 3 ou
   if(high_period/current_period<3)
     {
      Print("Erro! O período do indicador deve exceder o prazo atual de 3 ou mais veze
      return(false);
     }
//--- O período do indicador é o correto para a atual periodicidade
   return(true);
  }
//+------------------------------------------------------------------+
//| Receber dados de tempo a partir da maior prazo de tempo          |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- copiar todos os dados para o tempo atual
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- receber o código de erro
      int code=GetLastError();
      //--- imprimir a mensagem de erro
      PrintFormat("Erro de cópia de dados! %s",code==4401
                  ? "Histórico ainda está sendo carregado!"
                  : "Código = "+IntegerToString(code));
      //--- retornar falso para fazer uma tentativa de repetir o download de dados
      return(false);
     }
//--- recebe o tamanho do array
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir o índice do loop para o array zero
   ExtCount=0;
//--- definir a posição atual do candlestick no período de tempo para zero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- armazenar o último valor de tempo a partir da maior periodicidade
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Função constitui a parte principal do candlestick. Dependendo do sinalizador |
//| valor, a função define quais dados e arrays serão                            |
//| usados para exibição correta.                                                |
//+------------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- verificar o valor do flag
   if(flag)
     {
      //--- gerar o corpo do candlestick
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- gerar a sombra do candlestick
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- gerar o corpo do candlestick
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- gerar a sombra do candlestick
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| A função para formar o final do candlestick. Depende do valor do sinalizador, |
//| a função define quais dados e arrays são                                     |
//| usados para exibição correta.                                                |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- verificar o valor do flag
```

© 2000-2024, MetaQuotes Ltd.

```
   if(flag)
     {
      //--- gerar o final do corpo do candlestick
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- gerar o final da sombra do candlestick
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- alterar o valor do sinalizador para o oposto
      flag=false;
     }
   else
     {
      //--- gerar o final do corpo do candlestick
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- gerar o final da sombra do candlestick
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- alterar o valor do sinalizador para o oposto
      flag=true;
     }
  }
//+------------------------------------------------------------------------------------+
//| Limpar o fim do candlestick (a área entre a última e a penúltima                   |
//| barra)                                                                             |
//+------------------------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSec
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSec
  }
//+------------------------------------------------------------------------------+
//| Limpar o candlestick                                                         |
//+------------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- verificar
   if(count!=0)
     {
      //--- preencher buffers do indicador com valores vazios
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------------+
//| Gerar a parte principal do candlestick                                       |
//+------------------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- verificar
   if(count!=0)
     {
      //--- preencher o buffer do indicador com valores
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------+
//| Gerar o final do candlestick                                                 |
//+------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- preencher o buffer do indicador com valores
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Preencher i elemento dos buffers do indicador por valores vazios |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- definir um valor vazio na célula dos buffers do indicador
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayMinimum

Procura o elemento mais baixo na primeira dimensão de um array numérico multidimensional.

```
int  ArrayMinimum(
   const void&   array[],            // array para busca
   int           start=0,            // índice a partir do qual começa a verificação
   int           count=WHOLE_ARRAY   // número de elementos verificados
   );
```

Parâmetros

array[]

[in] Um array númerico, na qual a busca é feita.

start=0

[in] Índice a partir do qual começa a verificação.

count=WHOLE_ARRAY

[in] Número de elementos para busca. Por padrão, busca no array inteiro (count=WHOLE_ARRAY)..

Valor do Retorno

A função retorna um índice de um elemento encontrado levando em consideração a série do array. Em caso de falha, retorna -1.

Observação

O valor da flag AS_SERIES é levado em conta durante a busca de um mínimo.

Funções ArrayMaximum e ArrayMinimum aceitam arrays multi-dimensionais como um parâmetro. No entanto, a pesquisa é sempre aplicada à primeira (zero) dimensão.

Exemplo:

```
#property description "O indicador mostra sobre as barras (candlesticks) um prazo de t
//--- configurações do indicador
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- plot 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
```

© 2000-2024, MetaQuotes Ltd.

```
//---- plot 5
#property indicator_label5  "BullBody"
#property indicator_color5  clrOlive,clrOlive
//---- plot 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- plot 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- constante pré-definida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parâmetros de entrada
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Prazo de tempo para calcula
input datetime        InpDateStart=D'2013.01.01 00:00'; // Analisar a partir desta dat
//--- buffers do indicador para candlesticks de baixa
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- buffers do indicador para candlesticks de alta
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- variáveis globais
datetime ExtTimeBuff[];      // buffer de tempo para maior periodicidade
int      ExtSize=0;          // tamanho do buffer de tempo
int      ExtCount=0;         // índice dentro do buffer de tempo
int      ExtStartPos=0;      // posição inicial para cálculo do indicador
bool     ExtStartFlag=true;  // sinalizador auxiliar para receber a posição inicial
datetime ExtCurrentTime[1];  // último tempo na geração da barra de maior periodicidad
datetime ExtLastTime;        // último tempo a partir da barra de maior periodicidade,
bool     ExtBearFlag=true;   // flag para definir a ordem de escrita do dados para os
bool     ExtBullFlag=true;   // flag para definir a ordem de escrita do dados para os
int      ExtIndexMax=0;      // índice do elemento máximo do array
int      ExtIndexMin=0;      // índice do elemento mínimo do array
int      ExtDirectionFlag=0; // direção de movimento de preços para o atual candlestic
//--- alternar entre o preço de abertura e fechamento do candlestick para desenhar co
```

© 2000-2024, MetaQuotes Ltd.

```
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
//| Preenchendo a parte básica do candlestick                        |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- encontrar o índice dos elementos máximo e mínimo nos arrays
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // máximo em Alta
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // mínimo em Baixa
//--- definir quantas barras do período de tempo atual serão preenchidas
   int count=fill_index-start+1;
//--- se o preço de fechamento no primeiro bar é excede ao do último bar, o candlestic
   if(open[start]>close[last])
     {
      //--- antes que o candlestick seja de alta, limpar os valores de alta dos buffer
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- candlestick de baixa
      ExtDirectionFlag=-1;
      //--- gerar o candlestick
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- sair da função
      return;
     }
//--- se o preço de fechamento no primeiro bar é menor do que último bar, o candlestic
   if(open[start]<close[last])
     {
      //--- antes que o candlestick seja de baixa, limpar os valores de baixa dos buf
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- candlesticks de alta
      ExtDirectionFlag=1;
      //--- gerar o candlestick
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- sair da função
      return;
     }
//--- se você está nesta parte da função, o preço de abertura no primeiro bar é igual
//--- preço de fechamento no último bar; então o candlestick é considerado de baixa
//--- antes que o candlestick seja de alta, limpar os valores de alta dos buffers do
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- candlestick de baixa
   ExtDirectionFlag=-1;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se os preços de fechamento e de abertura são iguais, use o deslocamento para a e
   if(high[index_max]!=low[index_min])
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| Preencher o final do candlestick                                 |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- não desenhar em caso de uma única barra
   if(last-start==0)
      return;
//--- se o preço de fechamento no primeiro bar é excede ao do último bar, o candlestic
   if(open[start]>close[last])
     {
      //--- gerar o final do candelstick
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- sair da função
      return;
     }
//--- se o preço de fechamento no primeiro bar é menor do que último bar, o candlestic
   if(open[start]<close[last])
     {
      //--- gerar o final do candelstick
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- sair da função
      return;
     }
//--- se você está nesta parte da função, o preço de abertura no primeiro bar é igual
//--- preço de fechamento no último bar; então o candlestick é considerado de baixa
//--- gerar o final do candelstick
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEm
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificar o período do indicador
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- exibir dados de preço num primeiro plano
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- vincular buffers do indicador
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- definir alguns valores de propriedade para a criação do indicador
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // tipo de construção gráfic
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // estilo de linha de desenh
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // largura de linha de deser
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- neste caso ainda não existe cálculos de barras
   if(prev_calculated==0)
     {
      //--- receber a maior periodicidade das barra no tempo de chegada
      if(!GetTimeData())
         return(0);
     }
//--- definir a indexação direta
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- iniciar variável para o cálculo das barras
   int start=prev_calculated;
//--- se uma barra é gerada, recalcular o valor do indicador sobre ela
   if(start!=0 && start==rates_total)
      start--;
//--- o loop para cálculos dos valores do indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- preencher i elementos dos buffers do indicador por valores vazios
      FillIndicatorBuffers(i);
      //--- executar cálculos para as barras iniciando a partir dos dados InpDateStart
      if(time[i]>=InpDateStart)
        {
         //--- definir posição, a partir do qual os valores são exibidos pela primeira
         if(ExtStartFlag)
           {
            //--- armazenar o número do bar inicial
            ExtStartPos=i;
            //--- definir a primeira data a partir da maior periodicidade excedendo t
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- alterar o valor da sinalização em ordem não executada neste bloco n
            ExtStartFlag=false;
           }
         //--- verificar se ainda existem quaisquer elementos no array
         if(ExtCount<ExtSize)
           {
            //--- esperar o valor da periodicidade atual para chegar a um maior períod
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- desenhar a parte principal do candlestick (sem preencher o espaço
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- preencher o final do candlestick(a área entre a última e a penúlt
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtI
```

```
                //--- deslocar a posição inicial para desenhar o próximo candlestick
                ExtStartPos=i;
                //--- aumentar o contador de array
                ExtCount++;
               }
            else
                continue;
           }
         else
           {
            //--- redefinir os valores de array
            ResetLastError();
            //--- receber a última data do maior período de tempo
            if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
              {
               Print("Erro na cópia dos dados, código = ",GetLastError());
               return(0);
              }
            //--- se a nova data for posterior, parar de gerar o candlestick
            if(ExtCurrentTime[0]>ExtLastTime)
              {
               //--- limpar a área entre a última e a penúltima barra nos principais b
               ClearEndOfBodyMain(i-1);
               //--- preencher a área usando buffers auxiliares do indicador
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- deslocar a posição inicial para desenhar o próximo candlestick
               ExtStartPos=i;
               //--- restabelecer flag da direção do preço
               ExtDirectionFlag=0;
               //--- armazenar a nova última data
               ExtLastTime=ExtCurrentTime[0];
              }
            else
              {
               //--- gerar o candlestick
               FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Verificar a exatidão do período do indicador determinado         |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- o período do indicador deve ultrapassar o período de tempo que é exibido sobre
```

```
   if(current_period>=high_period)
     {
      Print("Erro! O valor do período do indicador deve exceder o valor do intervalo c
      return(false);
     }
//--- se o período do indicador é de uma semana ou um mês, o período está correto
   if(high_period>32768)
      return(true);
//--- converter valores do período para minutos
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- o período do indicador deve ser múltiplo do período de tempo, é exibido sobre
   if(high_period%current_period!=0)
     {
      Print("Erro! O valor do período do indicador deve ser múltiplo ao valor do atual
      return(false);
     }
//--- o período do indicador deve exceder o prazo de tempo quando apresentado de 3 ou
   if(high_period/current_period<3)
     {
      Print("Erro! O período do indicador deve exceder o prazo atual de 3 ou mais veze
      return(false);
     }
//--- O período do indicador é o correto para a atual periodicidade
   return(true);
  }
//+------------------------------------------------------------------+
//| Receber dados de tempo a partir da maior prazo de tempo          |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- copiar todos os dados para o tempo atual
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- receber o código de erro
      int code=GetLastError();
      //--- imprimir a mensagem de erro
      PrintFormat("Erro de cópia de dados! %s",code==4401
                  ? "Histórico ainda está sendo carregado!"
                  : "Código = "+IntegerToString(code));
      //--- retornar falso para fazer uma tentativa de repetir o download de dados
      return(false);
     }
//--- recebe o tamanho do array
   ExtSize=ArraySize(ExtTimeBuff);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir o índice do loop para o array zero
   ExtCount=0;
//--- definir a posição atual do candlestick no período de tempo para zero
   ExtStartPos=0;
   ExtStartFlag=true;
//--- armazenar o último valor de tempo a partir da maior periodicidade
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------------+
//| Função constitui a parte principal do candlestick. Dependendo do sinalizador |
//| valor, a função define quais dados e arrays serão                      |
//| usados para exibição correta.                                          |
//+------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- verificar o valor do flag
   if(flag)
     {
      //--- gerar o corpo do candlestick
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- gerar a sombra do candlestick
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- gerar o corpo do candlestick
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- gerar a sombra do candlestick
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| A função para formar o final do candlestick. Dependendo do valor da flag,  |
//| A função define quais dados e arrays são                                   |
//| usados para exibição correta.                                              |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- verificar o valor do flag
```

```
   if(flag)
     {
      //--- gerar o final do corpo do candlestick
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- gerar o final da sombra do candlestick
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- alterar o valor do sinalizador para o oposto
      flag=false;
     }
   else
     {
      //--- gerar o final do corpo do candlestick
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- gerar o final da sombra do candlestick
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- alterar o valor do sinalizador para o oposto
      flag=true;
     }
  }
//+------------------------------------------------------------------------------------
//| Limpar o final do candlestick (a área entre a última e a penúltima
//| barra)
//+------------------------------------------------------------------------------------
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSec
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSec
  }
//+------------------------------------------------------------------+
//| Limpar o candlestick                                             |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- verificar
   if(count!=0)
     {
      //--- preencher buffers do indicador com valores vazios
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| Gerar a parte principal do candlestick                           |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- verificar
   if(count!=0)
     {
      //--- preencher o buffer do indicador com valores
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Gerar o fim do candlestick                                       |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- preencher o buffer do indicador com valores
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Preencher i elemento dos buffers do indicador por valores vazios |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- definir um valor vazio na célula dos buffers do indicador
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayPrint

No log, exibe uma matriz de tipo ou estrutura simples.

```
void  ArrayPrint(
   const void&   array[],              // matriz de saída
   uint          digits=_Digits,       // número de casas decimais
   const string  separator=NULL,       // separador entre os valores dos campos de esti
   ulong         start=0,              // índice do primeiro elemento de saída
   ulong         count=WHOLE_ARRAY,    // número de elementos exibidos
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Parâmetros

array[]

[in] Matriz de tipo ou estrutura simples..

digits=_Digits

[in] Número de casas decimais para tipos reais. Por padrão igual a _Digits.

separator=NULL

[in] Separador entre os valores dos campos do elemento de estrutura. O valor padrão NULL indica que existe uma cadeia de caracteres vazia, neste caso, o separador é um espaço.

start=0

[in] Índice da primeira saída do elemento da matriz. Por padrão, exibe-se a partir do índice zero.

count=WHOLE_ARRAY

[in] Número de elementos de matriz que você deseja exibir. Por padrão, ele exibe toda a matriz (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] A combinação de sinalizadores especificada pelo modo de saída. Por padrão, todas os sinalizadores:

- ARRAYPRINT_HEADER - exibição dos cabeçalhos para a matriz das estruturas
- ARRAYPRINT_INDEX - exibição à esquerda do número de índice
- ARRAYPRINT_LIMIT - exibição somente dos 100 primeiros e os últimos 100 elementos da matriz. É usado quando você deseja exibir apenas parte de uma matriz maior.
- ARRAYPRINT_ALIGN - permitir o alinhamento dos valores de saída, quer dizer, os números serão alinhados à direita, enquanto as cadeias de caracteres à esquerda.
- ARRAYPRINT_DATE - ao exibir datetime, mostra a data no formato dd.mm.yyyy
- ARRAYPRINT_MINUTES - ao exibir datetime, mostra a data no formato HH:MM
- ARRAYPRINT_SECONDS - ao exibir datetime, mostra a data no formato HH:MM:SS

Valor de retorno

Não

Observação

ArrayPrint() não exibe, no log, todos os campos da matriz de estruturas, uma vez que os campos tanto de matriz como de indicadores de objetos são omitidos. Estas colunas simplesmente não serão

© 2000-2024, MetaQuotes Ltd.

exibidas para impressão, com o propósito de obter uma apresentação simples e conveniente. Se você precisar da exibição de todos os campos dessa estrutura, necessitará escrever sua própria função de saída em massa com a formatação desejada.

Exemplo:

```
//--- exibir os valores das últimas 10 barras
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Verificação\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spre
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("CopyRates failed, error code=%d",GetLastError());
//--- exemplo de saída
/*
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10   1
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   1
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   1
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   1
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   1
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    
   Verificação
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

Veja também

© 2000-2024, MetaQuotes Ltd.

FileSave, FileLoad

© 2000-2024, MetaQuotes Ltd.

ArrayRange

A função retorna o número de elementos na dimensão selecionada do array.

```
int  ArrayRange(
   const void&   array[],       // array para verificação
   int           rank_index     // índice da dimensão
   );
```

Parâmetros

array[]

[in] Array verificado.

rank_index

[in] Índice da dimensão.

Valor do Retorno

Número de elementos na dimensão selecionada do array.

Observação

Uma vez que os índices começam em zero, o número de dimensões de um array é uma unidade maior que o índice da última dimensão.

Exemplo:

```
void OnStart()
  {
//--- cria um array de quatro dimensões
   double array[][5][2][4];
//--- definir o tamanho da dimensão de zero
   ArrayResize(array,10,10);
//--- imprimir as dimensões
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- receber o tamanho de i dimensão
      temp=ArrayRange(array,i);
      //--- imprimir
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- Resultado
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrayResize

A função define um tamanho novo para a primeira dimensão

```
int  ArrayResize(
   void&  array[],              // array passado por referência
   int    new_size,             // novo tamanho de array
   int    reserve_size=0        // valor do tamanho de reserva (excesso)
   );
```

Parâmetros

array[]

[out] Array para alteração de tamanho.

new_size

[in] Novo tamanho para a primeira dimensão.

reserve_size=0

[in] Tamanho alocado para obter reserva.

Valor do Retorno

Se executado com sucesso, a função retorna a quantidade de todos os elementos contidos no array após a alteração do tamanho, caso contrário, retorna -1, e o tamanho do array não é alterado.

Если ArrayResize() применена к статическому массиву, таймсерии или индикаторному буферу, то размер массива остается прежним - такие массивы не могут быть перераспределены. В этом случае если *new_size*<=ArraySize(*array*), то функция просто вернет *new_size*; в противном случае будет возвращено -1.

Observação

A função pode ser aplicada somente para arrays dinâmicos. Deve ser notado que não se pode alterar o tamanho de arrays dinâmicos atribuídos como buffers de indicadores através da função SetIndexBuffer(). Para buffers de indicadores, todas a operações de redimensionamento de tamanho são realizadas pelo subsistema de tempo de execução do terminal.

Total amount of elements in the array cannot exceed 2147483647.

Com uma alocação de memória freqüente, é recomendado usar o terceiro parâmetro que defini um reserva para reduzir o número de alocações físicas de memória. Todas as subseqüentes chamadas de ArrayResize não levam uma realocação física de memória, mas somente alteração do tamanho da primeira dimensão do array dentro da memória reservada. Deve ser lembrado que o terceiro parâmetro será usado somente durante a alocação física de memória. Por exemplo:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

Neste caso a memória será realocada duas vezes, a primeira vez antes de entrar no loop de elemento 3000 (o tamanho do array será definido para 1000), e a segunda vez com i igual a 2000. Se omitisse o terceiro parâmetro, haveria 2000 realocações físicas de memória, que desaceleraria o programa.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Contadores
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- Um array para demonstração de uma versão rápida
   double arr[];
   ArrayResize(arr,100000,100000);
//--- Verifique quão rápida a variante com a reserva de memória trabalha
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- Define um novo tamanho de array especificando a reserva de 100 000 element
      ArrayResize(arr,i,100000);
      //--- Quando alcançar um número redondo, exibe o tamanho do array e tempo gasto
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-star
         start=now;
        }
     }
//--- Agora mostra quão lento a versão sem reserva de memória é
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Teste lenta: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- Define um novo tamanho de array, mas sem a reserva adicional
      ArrayResize(slow,i);
      //--- Quando alcançar um número redondo, exibe o tamanho do array e tempo gasto
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      }
   }
//--- O resultado do script
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

Também Veja

ArrayInitialize

© 2000-2024, MetaQuotes Ltd.

ArrayInsert

Insere na matriz de destino o número especificado de elementos da matriz de origem, iniciando no índice especificado.

```
bool  ArrayInsert(
   void&        dst_array[],          // matriz de destino
   const void&  src_array[],          // matriz de origem
   uint         dst_start,            // índice na matriz de destino para inserir
   uint         src_start=0,          // índice na matriz de origem para copiar
   uint         count=WHOLE_ARRAY     // número de elementos a serem inseridos
   );
```

Parâmetros

dst_array[]

[in][out] Matriz receptor à qual você deseja adicionar elementos.

src_array[]

[in] Matriz de origem a partir da qual você deseja adicionar elementos

dst_start

[in] Índice na matriz de destino para inserir elementos da matriz de origem.

src_start=0

[in] Índice na matriz de origem a partir do qual são tomados os elementos da matriz de origem para serem inseridos.

count=WHOLE_ARRAY

[in] Número de elementos adicionados a partir da matriz de origem. O valor WHOLE_ARRAY significa que serão inseridos todos os elementos do índice especificado até o final da matriz.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 5052 - ERR_SMALL_ARRAY (parâmetros *start* e/ou *count* inválidos ou matriz de origem *src_array[]* está vazia),
- 5056 - ERR_SERIES_ARRAY (a matriz não pode ser alterada, buffer de indicador),
- 4006 - ERR_INVALID_ARRAY (a cópia para si próprio é inválida, ou as matrizes têm um tipo diferente ou possuem uma matriz de tamanho fixo, que contém objetos de uma classe ou de uma estrutura com um destruidor),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (a matriz não contém estruturas POD, isto é, é impossível copiar),
- erros de redimensionamento da matriz de destino *dst_array[]* - eles são dados na descrição da função ArrayRemove().

Observação

Se a função é usada para uma matriz de tamanho fixo, o tamanho da própria matriz de destino *dst_array[]* não muda, a partir da posição *dst_start* os elementos da matriz de destino são

© 2000-2024, MetaQuotes Ltd.

deslocados para a direita (o último *count* dos elementos cai), e os elementos são copiados da matriz de origem para o local liberado.

Você não pode inserir elementos em matrizes dinâmicas, que são designadas como buffers de indicador, através da função SetIndexBuffer(). Para buffers de indicador, todas as operações de redimensionamento são executadas pelo subsistema em execução do terminal.

Na matriz de origem, os elementos são copiados a partir do índice *src_start*. O tamanho da matriz de origem não é alterado. Os elementos adicionados à matriz de destino não são referências aos elementos da matriz de origem, o que significa que alterações subsequentes nos elementos em qualquer uma das matrizes não são refletidas na seguinte.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declara uma matriz de tamanho fixo e preenche com valores
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- matriz de origem
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- mostra matrizes antes de inserir elementos
   Print("Antes de chamar a ArrayInsert()");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- insere 3 elementos da matriz de origem e mostra a nova composição da matriz de d
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("Depois de chamar a ArrayInsert()");
   ArrayPrint(array_dest);
/*
  Resultado de execução
   Antes de chamar a ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   Depois de chamar aArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

Veja também

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

ArrayRemove

Remove o número especificado de elementos da matriz, iniciando no índice especificado.

```
bool  ArrayRemove(
   void&        array[],            // matriz de qualquer tipo
   uint         start,              // índice a partir do qual é iniciada a remoção
   uint         count=WHOLE_ARRAY   // número de elementos
   );
```

Parâmetros

array[]

[in][out] Matriz.

start

[in] Índice a partir do qual são excluídos os elementos da matriz.

count=WHOLE_ARRAY

[in] Número de elementos para excluir. O valor WHOLE_ARRAY indica a remoção de todos os elemento, do índice especificado até o final da matriz.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 5052 - ERR_SMALL_ARRAY (o valor *start* é muito grande),
- 5056 - ERR_SERIES_ARRAY (a matriz não pode ser alterada, buffer de indicador),
- 4003 - ERR_INVALID_PARAMETER (o valor *count* é muito grande),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (matriz de tamanho fixo que contém objetos complexos com um destruidor),
- 4006 - ERR_INVALID_ARRAY (matriz de tamanho fixo que contém objetos de estruturas ou de classes com destruidores).

Observação

Se a função é usada para uma matriz de tamanho fixo, o tamanho da matriz em si não muda: neste caso, a "cauda" remanescente é fisicamente copiada para a posição *start*. Para um entendimento preciso de como funciona a função, veja o exemplo abaixo. Cópia "física" significa que os objetos copiados não são criados chamando o construtor ou o operador de cópia, mas, sim, simplesmente copiando a representação binária do objeto. É por esse motivo que é proibido aplicar a função ArrayRemove() à uma matriz de tamanho fixo contendo objetos com um destruidor (é ativado ERR_INVALID_ARRAY ou ERR_STRUCT_WITHOBJECTS_ORCLASS) Como ao remover tal objeto, o destruidor deve ser chamado duas vezes - para o objeto inicial e para sua cópia.

É impossível excluir elementos de matrizes dinâmicas, atribuídas como buffers de indicador, através da função SetIndexBuffer(), isso resultará no erro ERR_SERIES_ARRAY. Para buffers de indicador, todas as operações de redimensionamento são executadas pelo subsistema em execução do terminal.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declara uma matriz de tamanho fixo e preenche com valores
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostra uma matriz antes de remover elementos
   Print("Antes de chamar a ArrayRemove()");
   ArrayPrint(array);
//--- remove 2 elementos da matriz e mostra a nova composição
   ArrayRemove(array,4,2);
   Print("Depois de chamar a ArrayRemove()");
   ArrayPrint(array);
/*
  Resultado da execução:
  Antes de chamar a ArrayRemove()
   0 1 2 3 4 5 6 7 8 9
  Depois de chamar a ArrayRemove()
   0 1 2 3 6 7 8 9 8 9
*/
```

Veja também

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

© 2000-2024, MetaQuotes Ltd.

ArrayReverse

Expande o número de elementos especificado na matriz, iniciando no índice especificado.

```
bool  ArrayReverse(
   void&        array[],            // matriz de qualquer tipo
   uint         start=0,            // índice a partir do qual é expandida a matriz
   uint         count=WHOLE_ARRAY   // número de elementos
   );
```

Parâmetros

array[]

[in][out] Matriz.

start=0

[in] O índice a partir do qual a matriz é invertida.

count=WHOLE_ARRAY

[in] Número de elementos para expandir. Se for especificado WHOLE_ARRAY, todos os elementos da matriz serão colocados de maneira invertida entre eles, começando com o índice especificado *start* até o final da matriz.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Observação

A função ArraySetAsSeries() apenas altera a direção de indexação de trás para frente, a fim de organizar o acesso aos elementos como em timeseries. A função ArrayReverse() move fisicamente os elementos de uma matriz de tal forma que a matriz é "invertida".

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declara uma matriz de tamanho fixo e preenche com valores
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- mostra uma matriz antes de inversão de elementos
   Print("Antes de chamar a ArrayReverse()");
   ArrayPrint(array);
//--- inverte 3 elementos na matriz e exibe a nova composição
   ArrayReverse(array,4,3);
   Print("Depois de chamar a ArrayReverse()");
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayPrint(array);
/*
  Resultado da execução:
  Antes de chamar a ArrayReverse()
   0 1 2 3 4 5 6 7 8 9
  Depois de chamar a ArrayReverse()
   0 1 2 3 6 5 4 7 8 9
*/
```

Veja também

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArraySetAsSeries

A função define o flag AS_SERIES para um objeto de um array dinâmico, e os elementos serão indexados como em timeseries.

```
bool  ArraySetAsSeries(
   const void&  array[],     // array passado por referência
   bool         flag         // true significa ordem reversa de indexação
   );
```

Parâmetros

array[]

[in][out] Array numérico para definir.

flag

[in] Direção de indexação do array.

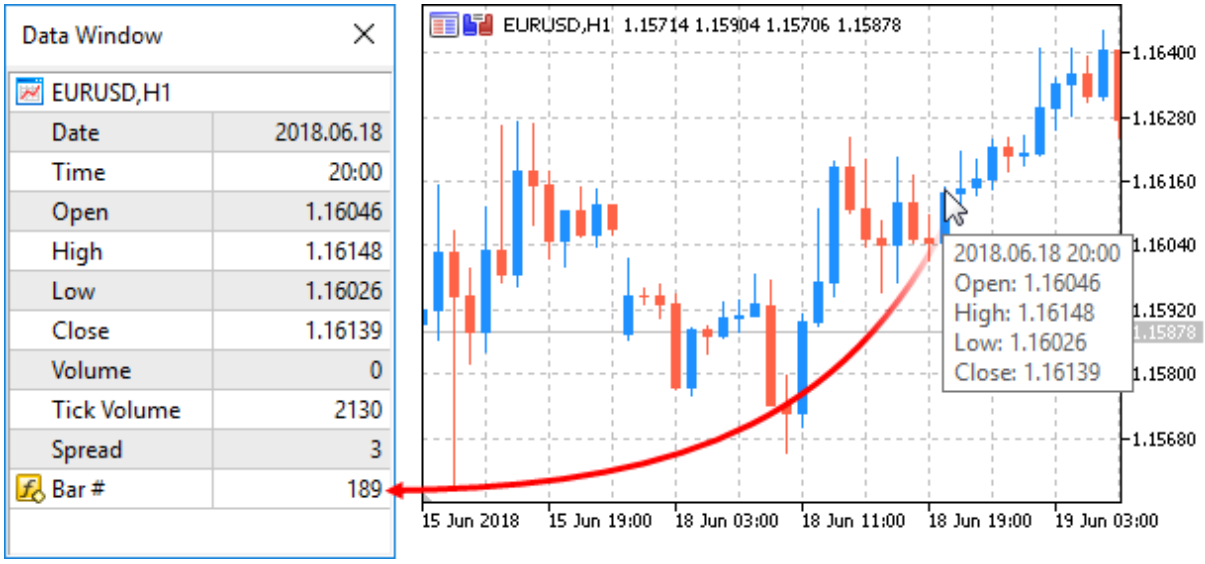
Valor do Retorno

A função retorna true para sucesso, caso contrário - false.

Observação

O flag AS_SERIES não pode ser definido para arrays multi-dimensionais ou arrays estáticos (arrays cujo tamanho nos colchetes já é predefinido na etapa de compilação). A indexação em séries de tempo difere da indexação de um array comum, na medida que os elementos de séries de tempo são indexados do fim para o começo (dos dados mais novos para os mais antigos).

Exemplo: Indicador que mostra número de barra




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plota Numeration
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_label1  "Numeration"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- buffers do indicador
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- definir a indexação do buffer como numa timeseries
   ArraySetAsSeries(NumerationBuffer,true);
//--- define a precisão de exibição na Janela de Dados
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- como o nome do array do indicador é exibido na Janela de Dados
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- nós armazenaremos a hora de abertura da barra zero corrente
   static datetime currentBarTimeOpen=0;
//--- acesso reverso ao array time[] - fazemos isso como em séries temporais
   ArraySetAsSeries(time,true);
//--- se a hora da barra zero difere da hora armazenada
   if(currentBarTimeOpen!=time[0])
     {
     //--- enumera todas as barras a partir do corrento para a profundidade do gráfico
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

Também Veja

Acesso à séries de tempo, ArrayGetAsSeries

© 2000-2024, MetaQuotes Ltd.

ArraySize

A função retorna o número de elementos de um array selecionado.

```
int  ArraySize(
   const void&  array[]    // array verificado
   );
```

Parâmetros

array[]

[in] Array de qualquer tipo.

Valor do Retorno

Valor do tipo int.

Observação

Para um array de uma dimensão, o valor a ser retornado pela função ArraySize é igual ao de ArrayRange(array,0).

Exemplo:

```
void OnStart()
  {
//--- criar arrays
   double one_dim[];
   double four_dim[][10][5][2];
//--- tamanhos
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- variáveis auxiliares
   int size;
//--- alocar memória sem fazer backup
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. array unidimensional
   Print("+==============================================================+");
   Print("Tamanhos de arrays:");
   Print("1. Array unidimensional");
   size=ArraySize(one_dim);
   PrintFormat("Tamanho dimensão zero = %d, Tamanho array = %d",one_dim_size,size);
//--- 2. array multidimensional
   Print("2. Array multidimensional");
   size=ArraySize(four_dim);
   PrintFormat("Tamanho dimensão zero = %d, Tamanho array = %d",four_dim_size,size);
//--- tamanhos de dimensões
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("Verificar:");
   Print("Dimensão zero = Tamanho de array / (Primeira dimensão * Segunda dimensão * T
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. array unidimensional com backup de memória
   Print("3. Array unidimensional com backup de memória");
//--- dobro do valor
   one_dim_size*=2;
//--- alocar memória com o backup
   ArrayResize(one_dim,one_dim_size,reserve);
//--- imprimir o tamanho
   size=ArraySize(one_dim);
   PrintFormat("Tamanho com o backup = %d, Atual tamanho do array = %d",one_dim_size+r
  }
```

© 2000-2024, MetaQuotes Ltd.

ArraySort

Classifica os valores na primeira dimensão de um array numérico multidimensional na ordem crescente.

```
bool  ArraySort(
   void&  array[]      // array para ordenação
   );
```

Parâmetros

array[]

[in][out] Array numérico para ordenação.

Valor do Retorno

A função retorna true para sucesso, caso contrário - false.

Observação

Um array é sempre classificado em ordem crescente, independentemente do valor da flag AS_SERIES.

Funções ArraySort e ArrayBSearch aceitam arrays de qualquer dimensão como um parâmetro. No entanto, a pesquisa e a classificação são sempre aplicadas à primeira dimensão (zero).

Exemplo:

```
#property description "O indicador analisa os dados para o último mês e pinta todos os
#property description "e grandes volumes de tick. A volume de tick do array é ordenado
#property description "para definir estes candlesticks. Os candlesticks que tem os vol
#property description "por cento do array, são consideradas pequenos. Os candlesticks
#property description "por cento do array, são consideradas grandes."
//--- configurações do indicador
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plotar
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- constante pré-definida
#define INDICATOR_EMPTY_VALUE 0.0
//--- parâmetros de entrada
input int InpSmallVolume=15; // Valor percentual de pequenos volumes (<50)
input int InpBigVolume=20;   // Valor percentual de grandes volumes (<50)
//--- início do tempo para análise (será deslocado)
datetime ExtStartTime;
//--- buffers do indicador
double   ExtOpenBuff[];
```

© 2000-2024, MetaQuotes Ltd.

```
double   ExtHighBuff[];
double   ExtLowBuff[];
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- valores de limite do volume para exibir os candlesticks
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| Receber valores das bordas para volumes de tick                  |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- variáveis
   datetime stop_time;  // copiar tempo final
   long     buff[];     // buffer para copiar
//--- tempo final é o atual
   stop_time=TimeCurrent();
//--- tempo de início é de um mês antes do atual
   ExtStartTime=GetStartTime(stop_time);
//--- receber os valores de volume do tick
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- falhou para receber os dados, retornar falso para lançar o comando de reca
      PrintFormat("Falha ao receber os valores de volume do tick. Código de erro = %d"
      return(false);
     }
//--- calcule tamanho de array
   int size=ArraySize(buff);
//--- classificar o array
   ArraySort(buff);
//--- definir os valores das bordas esquerda e direita para volumes do tick
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- sucesso na execução
   return(true);
  }
//+------------------------------------------------------------------+
//| Receber os dados do mês menor do que o mês anterior              |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- converter o tempo final da variável de estrutura do tipo MqlDateTime
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- receber os dados que é do mês menor
   if(temp.mon>1)
      temp.mon-=1;  // o mês corrente não é o primeiro no ano, por conseguinte, o núme
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       temp.mon=12;  // o mês corrente é o primeiro no ano, por conseguinte, o número d
       temp.year-=1; // enquanto o número do ano é um a menos
      }
//--- número do dia não será superior a 28
   if(temp.day>28)
       temp.day=28;
//--- retornar a data obtida
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificar se os parâmetros de entrada satisfazem as condições
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Parâmetros de entrada incorreto");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- definir o valor que não será exibido
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- definir etiquetas para buffers do indicador
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- verificar se barras não manipuladas ainda estão presentes
```

© 2000-2024, MetaQuotes Ltd.

```
   if(prev_calculated<rates_total)
     {
      //--- receber novos valores das bordas direita e esquerda para volumes
      if(!GetVolumeBorders())
         return(0);
     }
//--- iniciar variável para cálculo da barra
   int start=prev_calculated;
//--- trabalho na última barra se os valores do indicador já foram calculados no tick
   if(start>0)
      start--;
//--- definir indexação direta em séries temporais
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- o loop de cálculo dos valores do indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- preencher os candlesticks a partir da data inicial
      if(ExtStartTime<=time[i])
        {
         //--- se o valor não é inferior a borda direita, preencher clandlestick
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- receber dados para desenhar o candlestick
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- cor de DodgerBlue
            ExtColorBuff[i]=0;
            //--- continuar o loop
            continue;
           }
         //--- preencher o candlestick, se o valor não exceder o limite esquerdo
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- receber dados para desenhar o candlestick
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- Cor laranja
            ExtColorBuff[i]=1;
            //--- continuar o loop
            continue;
```

© 2000-2024, MetaQuotes Ltd.

```
            }
         }
      //--- definir valores vazios para as barras que não foram incluídas no cálculo
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

ArrayBsearch

© 2000-2024, MetaQuotes Ltd.

ArraySwap

Troca entre si o conteúdo de duas matrizes do tipo dinâmico. Para matrizes multidimensionais, o número de elementos em todas as dimensões, excepto o primeiro, tem de ser o mesmo.

```
bool  ArraySwap(
   void&  array1[],      // primeira matriz
   void&  array2[]       // segunda matriz
   );
```

Parâmetros

array1[]

[in][out] Matriz numérica.

array2[]

[in][out] Matriz numérica.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false. Neste caso, GetLastError() retorna o código de erro ERR_INVALID_ARRAY.

Observação

A função implementa matrizes dinâmicas do mesmo tipo e as mesmas dimensões, exceto a primeira. Para tipos inteiros, o sinal é ignorado, ou seja, char==uchar)

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- matrizes para armazenar cotações
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- obtemos dados dos últimos 20 candles no timeframe atual
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- definimos o tamanho da matriz pelo número de dados copiados
   ArrayResize(source_array,copied);
//--- preenchemos a matriz rate_array_1[] com dados a partir de rates[]
   for(int i=0;i<copied;i++)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      source_array[i][0]=(double)rates[i].time;
      source_array[i][1]=rates[i].open;
      source_array[i][2]=rates[i].high;
      source_array[i][3]=rates[i].low;
      source_array[i][4]=rates[i].close;
      source_array[i][5]=(double)rates[i].tick_volume;
      source_array[i][6]=(double)rates[i].spread;
      source_array[i][7]=(double)rates[i].real_volume;
     }
//--- realizamos o intercâmbio de dados entre source_array[] e dest_array[]
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%d",GetLast
      return;
     }
//--- verificamos que, depois do intercâmbio, o tamanho da fonte-matriz se tornou zero
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- exibimos os dados do receptor de matriz dest_array[]
   ArrayPrint(dest_array);
  }
```

Veja também

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

© 2000-2024, MetaQuotes Ltd.

Matrizes e vetores

Uma matriz é um array bidimensional de números do tipo double, float ou complex.

Um vetor é uma matriz unidimensional de números do tipo double, float ou complex. O vetor não tem indicação de se é vertical ou horizontal. Essa informação é sabido a partir do contexto de uso, por exemplo, a operação vetorial Dot assume que o vetor esquerdo é horizontal, o vetor direito é vertical. Se for necessária uma precisão absoluta, podem ser usadas matrizes de uma única linha ou coluna, mas geralmente isto não é necessário.

Matrizes e vetores alocam memória para dados dinamicamente. Na verdade, matrizes e vetores são objetos que possuem certas propriedades, como o tipo de dados que contêm e as dimensões. Propriedades de matrizes e vetores podem ser obtidas usando métodos como, vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() e por aí fora. E a dimensão pode ser alterada.

Ao criar e inicializar matrizes, são utilizados os chamados métodos estáticos (estes são como os métodos de classe estática), por exemplo: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

No momento, as operações sobre matrizes e vetores não envolvem o uso de um tipo de dado complexo - este ainda é um trabalho em andamento.

MQL5 suporta a passagem de matrizes e vetores para DLLs. Assim, funções que usam esse tipo de dados são importadas a partir de bibliotecas externas.

Matrizes e vetores são passados para DLL como um ponteiro para um buffer. Por exemplo, para passar uma matriz do tipo float, o parâmetro correspondente da função DLL exportada deve levar um ponteiro para um buffer do tipo float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

Para um processamento adequado de matrizes e vetores, além de seus buffers, devem ser passados seus tamanhos.

Todos os métodos matriciais e vetoriais estão listados abaixo em ordem alfabética.

Função	Ação	Categoria
Activation	Calcula os valores da função de ativação e registra no vetor/matriz passado	Aprendizado de máquina
ArgMax	Retorna o índice do valor máximo	Estatísticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
ArgMin	Retorna o índice do valor mínimo	Estatísticas
ArgSort	Retorna o índice classificado	Atribuições
Assign	Copia uma matriz, vetor ou array com conversão automática	Inicialização
Average	Calcula a média ponderada dos valores da matriz/vetor	Estatísticas
Cholesky	Calcula a decomposição de Cholesky	Conversões
Clip	Limita os elementos de uma matriz/vetor a um determinado intervalo de valores válidos	Atribuições
Col	Retorna um vetor coluna. Grava um vetor na coluna especificada	Atribuições
Cols	Retorna o número de colunas em uma matriz	Características
Compare	Compara os elementos de duas matrizes/vetores com uma determinada precisão	Atribuições
CompareByDigits	Compara se os elementos de duas matrizes/vetores correspondem, levando em conta dígitos significativos	Atribuições
Cond	Calcula o número condicional de uma matriz	Características
Convolve	Retorna a convolução linear discreta de dois vetores	Produtos
Copy	Retorna uma cópia da matriz/vetor fornecida	Atribuições
CopyIndicatorBuffer	Obtém no vetor dados do buffer especificado do indicador especificado com a quantidade indicada	Inicialização
CopyRates	Obtém em uma matriz ou vetor a série histórica da estrutura MqlRates do período-símbolo especificado na quantidade especificada	Inicialização

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
CopyTicks	Obtém na matriz ou vetor os ticks a partir da estrutura MqlTick	Inicialização
CopyTicksRange	Obtém na matriz ou vetor os ticks a partir da estrutura MqlTick dentro do intervalo de datas especificado	Inicialização
CorrCoef	Calcula o coeficiente de correlação de Pearson (coeficiente de correlação linear)	Produtos
Correlate	Calcula a correlação cruzada de dois vetores	Produtos
Cov	Calcula a matriz de covariância	Produtos
CumProd	Retorna o produto cumulativo dos elementos de uma matriz/vetor, incluindo os elementos ao longo do eixo dado	Estatísticas
CumSum	Retorna a soma cumulativa dos elementos de uma matriz/vetor, incluindo elementos ao longo de um determinado eixo	Estatísticas
Derivative	Calcula os valores da derivada da função de ativação e escreve no vetor/matriz passado	Aprendizado de máquina
Det	Calcula o determinante de uma matriz quadrada não degenerada	Características
Diag	Extrai uma diagonal ou constrói uma matriz diagonal	Atribuições
Dot	Produto escalar de dois vetores	Produtos
Eig	Calcula os valores próprios e os vetores próprios direitos de uma matriz quadrada	Conversões
EigVals	Calcula os valores próprios de uma matriz comum	Conversões
Eye	Retorna uma matriz com uns na diagonal e zeros nos outros lugares	Inicialização

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
Fill	Preenche uma matriz ou vetor existente com o valor fornecido	Inicialização
Flat	Permite acessar um elemento da matriz por meio de um índice em vez de dois	Atribuições
Full	Cria e retorna uma nova matriz preenchida com o valor fornecido	Inicialização
GeMM	Produto matricial geral de duas matrizes (General Matrix Multiply)	Produtos
HasNan	Retorna o número de valores NaN na matriz/vetor	Atribuições
Hsplit	Divisão horizontal de uma matriz em várias submatrizes. O mesmo que Split com axis=0	Atribuições
Identity	Cria uma matriz de identidade do tamanho especificado	Inicialização
Init	Inicialização de matriz ou vetor	Inicialização
Inner	Produto interno de duas matrizes	Produtos
Inv	Calcula o inverso (multiplicativo) de uma matriz quadrada não degenerada usando o método Gauss-Jordan	Soluções
Kron	Retorna o produto Kronecker de duas matrizes, uma matriz e um vetor, um vetor e uma matriz ou dois vetores	Produtos
LinearRegression	Calcula o vetor/matriz com os valores de regressão linear calculados	Estatísticas
Loss	Calcula os valores da função de perda e grava no vetor/matriz passado	Aprendizado de máquina
LstSq	Retorna a solução de mínimos quadrados de equações algébricas lineares (para matrizes não quadradas ou singulares)	Soluções

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
LU	Decomposição LU de uma matriz como um produto de uma matriz triangular inferior e uma matriz triangular superior	Conversões
LUP	Fatorização LUP com permutação parcial, que se refere à decomposição LU apenas com permutação de linhas: PA=LU	Conversões
MatMul	Produto de matriz de duas matrizes	Produtos
Max	Retorna o valor máximo em uma matriz/vetor	Estatísticas
Mean	Calcula a média aritmética dos valores dos elementos	Estatísticas
Median	Calcula a mediana dos elementos de uma matriz/vetor	Estatísticas
Min	Retorna o valor mínimo em uma matriz/vetor	Estatísticas
Norm	Retorna a norma de uma matriz ou vetor	Características
Ones	Cria e retorna uma nova matriz preenchida com uns	Inicialização
Outer	Calcula o produto externo de duas matrizes ou dois vetores	Produtos
Percentile	Retorna o percentil especificado de valores de elemento de matriz/vetor ou elementos ao longo do eixo especificado	Estatísticas
PInv	Calcula o pseudoinverso pelo método Moore-Penrose	Soluções
Power	Eleva uma matriz quadrada a uma potência inteira	Produtos
Prod	Retorna o produto dos elementos da matriz/vetor, que também pode ser realizado para o eixo dado	Estatísticas
Ptp	Retorna o intervalo de valores de uma matriz/vetor ou um determinado eixo de uma matriz	Estatísticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
QR	Calcula a fatoração qr de uma matriz	Conversões
Quantile	Retorna o quantil especificado dos valores dos elementos da matriz/vetor ou dos elementos ao longo do eixo especificado	Estatísticas
Rank	Retorna o posto de uma matriz usando o método de Gauss	Características
RegressionMetric	Calcula a métrica de regressão como o erro de desvio da linha de regressão criada na matriz de dados especificada	Estatísticas
Reshape	Alteração da forma de uma matriz sem alterar seus dados	Atribuições
Resize	Retorna uma nova matriz com forma e tamanho modificados	Atribuições
Row	Retorna um vetor linha. Grava um vetor na string especificada	Atribuições
Rows	Retorna o número de linhas em uma matriz	Características
Set	Define o valor de um elemento vetorial pelo índice especificado	Atribuições
Size	Retorna o tamanho do vetor	Características
Set	Define o valor de um elemento vetorial pelo índice especificado	Atribuições
SLogDet	Calcula o sinal e o logaritmo do determinante de uma matriz	Características
Solve	Resolve uma equação de matriz linear ou um sistema de equações algébricas lineares	Soluções
Sort	Classificação por lugar	Atribuições
Spectrum	Calcula o espectro de uma matriz como o conjunto de seus autovalores do produto AT*A	Características
Split	Divisão de uma matriz em várias submatrices	Atribuições
Std	Retorna o desvio padrão dos valores dos elementos de uma	Estatísticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Categoria
	matriz/vetor ou elementos ao longo de um determinado eixo	
Sum	Retorna a soma dos elementos da matriz/vetor, o que também pode ser feito para o(s) eixo(s) especificado(s)	Estatísticas
SVD	Decomposição em valores singulares	Conversões
SwapCols	Troca colunas em uma matriz	Atribuições
SwapRows	Troca linhas em uma matriz	Atribuições
Trace	Retorna a soma das diagonais de uma matriz	Características
Transpose	Transpõe (inverte os eixos) e retorna a matriz modificada	Atribuições
Tri	Constrói uma matriz com uns em uma determinada diagonal e abaixo, e zeros nos outros lugares	Inicialização
TriL	Retorna uma cópia da matriz zerada acima da k-ésima diagonal. Matriz triangular inferior	Atribuições
TriU	Retorna uma cópia da matriz com elementos zerados abaixo da k-ésima diagonal Matriz triangular superior	Atribuições
Var	Calcula a variância dos valores dos elementos de uma matriz/vetor	Estatísticas
Vsplit	Divisão vertical de uma matriz em várias submatrizes. O mesmo que Split com axis=1	Atribuições
Zeros	Cria e retorna uma nova matriz preenchida com zeros	Inicialização

© 2000-2024, MetaQuotes Ltd.

Tipos de matrizes e vetores

matrix и vector — são tipos de dados especiais em MQL5 projetados para realizar operações a partir de álgebra linear. Existem os seguintes tipos de dados:

- matrix — matriz contendo elementos do tipo double
- matrixf — matriz contendo elementos do tipo float
- matrixc — matriz contendo elementos do tipo complex.
- vector — vetor contendo elementos do tipo double
- vectorf — vetor contendo elementos do tipo float
- vectorc — vetor contendo elementos do tipo complex

Para uso em funções de modelo, é possível usar a entrada matrix<double>, matrix<float>, matrix<complex>, vector<double>, vector<float>, vector<complex> em vez dos tipos correspondentes.

MQL5 suporta a passagem de matrizes e vetores para DLLs. Assim, funções que usam esse tipo de dados são importadas a partir de bibliotecas externas.

Matrizes e vetores são passados para DLL como um ponteiro para um buffer. Por exemplo, para passar uma matriz do tipo float, o parâmetro correspondente da função DLL exportada deve levar um ponteiro para um buffer do tipo float.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

Para um processamento adequado de matrizes e vetores, além de seus buffers, devem ser passados seus tamanhos.

Métodos de inicialização de matrizes e vetores

Função	Ação
Assign	Copia uma matriz, vetor ou array com conversão automática
CopyRates	Obtém em uma matriz ou vetor a série histórica da estrutura MqlRates do período-símbolo especificado na quantidade especificada
Eye	Retorna uma matriz com uns na diagonal principal e zeros em todas as outras posições.
Identity	Cria uma matriz de identidade do tamanho especificado
Ones	Cria e retorna uma nova matriz preenchida com uns
Zeros	Cria e retorna uma nova matriz preenchida com zeros

© 2000-2024, MetaQuotes Ltd.

Função	Ação
Full	Cria e retorna uma nova matriz preenchida com os valores fornecidos
Tri	Cria uma matriz com uns na diagonal indicada e abaixo e zeros em todas as outras posições
Init	Inicializa uma matriz ou vetor
Fill	Preenche uma matriz ou vetor existente com o valor fornecido

© 2000-2024, MetaQuotes Ltd.

Enumerações para manusear matrizes e vetores

Esta seção descreve as enumerações que são usadas nos vários métodos matriciais e vetoriais.

ENUM_AVERAGE_MODE

Enumeração de tipos de cálculo de média.

Identificador	Descrição
AVERAGE_NONE	Sem média. Os resultados são fornecidos para cada rótulo separadamente
AVERAGE_BINARY	Resultado do marcador 1 para classificação binária
AVERAGE_MICRO	Resultado da média da matriz de erros (confusion matrix)
AVERAGE_MACRO	Resultado médio dos resultados das matrizes de erro de cada rótulo
AVERAGE_WEIGHTED	Resultado médio ponderado

ENUM_VECTOR_NORM

Enumeração de normas vetoriais para vector::Norm.

Identificador	Descrição
VECTOR_NORM_INF	Norma Infinita
VECTOR_NORM_MINUS_INF	Norma infinita negativa
VECTOR_NORM_P	Norma-p

ENUM_MATRIX_NORM

Enumeração de normas matriciais para matrix::Norm e para obter o número de condicionalidade da matriz matrix::Cond.

Identificador	Descrição
MATRIX_NORM_FROBENIUS	norma de Frobenius
MATRIX_NORM_SPECTRAL	Norma espectral
MATRIX_NORM_NUCLEAR	Norma nuclear
MATRIX_NORM_INF	Norma Infinita
MATRIX_NORM_P1	Norma P1

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
MATRIX_NORM_P2	Norma P2
MATRIX_NORM_MINUS_INF	Norma infinita negativa
MATRIX_NORM_MINUS_P1	Norma negativa P1
MATRIX_NORM_MINUS_P2	Norma negativa P2

ENUM_VECTOR_CONVOLVE

Enumeração para a convolução vector::Convolve e correlação cruzada vector::Correlate.

Identificador	Descrição
VECTOR_CONVOLVE_FULL	Convolução completa
VECTOR_CONVOLVE_SAME	Convolução com tipo same
VECTOR_CONVOLVE_VALID	Convolução com tipo valid

ENUM_REGRESSION_METRIC

Enumeração de métricas de regressão para vector::RegressionMetric.

Identificador	Descrição
REGRESSION_MAE	Erro absoluto médio
REGRESSION_MSE	Erro quadrático médio
REGRESSION_RMSE	Raiz quadrada do erro
REGRESSION_R2	R-quadrado
REGRESSION_MAPE	Erro absoluto médio em porcentagem
REGRESSION_MSPE	Erro quadrático médio em porcentagem
REGRESSION_RMSLE	Raiz do erro quadrático médio logarítmico
REGRESSION_SMAPE	Erro absoluto médio simétrico em porcentagem
REGRESSION_MAXE	Erro absoluto máximo
REGRESSION_MEDE	Erro absoluto mediano
REGRESSION_MPD	Desvio médio de Poisson
REGRESSION_MGD	Desvio gama médio
REGRESSION_EXPV	Variância explicada

© 2000-2024, MetaQuotes Ltd.

ENUM_CLASSIFICATION_METRIC

Enumeração de métricas para tarefas de classificação.

Identificador	Descrição
CLASSIFICATION_ACCURACY	Qualidade do modelo em termos de fidelidade das previsões em todas as classes
CLASSIFICATION_AVERAGE_PRECISION	Precisão média do modelo
CLASSIFICATION_BALANCED_ACCURACY	Precisão de previsões balanceada
CLASSIFICATION_F1	Medida-F1. Média harmônica entre precisão (precision) e recuperação (recall) do modelo
CLASSIFICATION_JACCARD	Índice Jaccard
CLASSIFICATION_PRECISION	Precisão do modelo na previsão de verdadeiros positivos para a classe-alvo
CLASSIFICATION_RECALL	Integridade do modelo
CLASSIFICATION_ROC_AUC	Área sob a curva de erro
CLASSIFICATION_TOP_K_ACCURACY	Frequência de ocorrência do rótulo correto no topo dos rótulos K previstos

ENUM_LOSS_FUNCTION

Enumeração para o cálculo da função de perda vector::Loss.

Identificador	Descrição
LOSS_MSE	Erro quadrático médio
LOSS_MAE	Erro absoluto médio
LOSS_CCE	Entropia cruzada categórica
LOSS_BCE	Entropia cruzada binária
LOSS_MAPE	Erro absoluto médio em porcentagem
LOSS_MSLE	Erro logarítmico quadrático médio
LOSS_KLD	Divergência de Kullback-Leibler
LOSS_COSINE	Semelhança/proximidade de cosseno
LOSS_POISSON	Função de perda de Poisson
LOSS_HINGE	Função de perda linear por partes (Hinge loss)

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
LOSS_SQ_HINGE	Função de perda linear por partes quadrática
LOSS_CAT_HINGE	Função de perda linear por partes categórica
LOSS_LOG_COSH	Logaritmo de cosseno hiperbólico
LOSS_HUBER	Função de perda de Huber

ENUM_ACTIVATION_FUNCTION

Enumeração para a função de ativação vector::Activation e a derivada da função de ativação vector::Derivative.

Identificador	Descrição	Parâmetros
AF_NONE	A função de ativação não é usada, o valor na entrada é passado para a saída	
AF_ELU	Unidade linear exponencial	
AF_EXP	Exponencial	
AF_GELU	Unidade linear gaussiana de erro	
AF_HARD_SIGMOID	Sigmóide rígido	
AF_LINEAR	Linear	
AF_LRELU	Retificador linear com "vazamento" (Leaky ReLU)	
AF_RELU	Transformação linear truncada ReLU	
AF_SELU	Função linear exponencial escalada (Scaled ELU)	
AF_SIGMOID	Sigmoide	
AF_SOFTMAX	Softmax	
AF_SOFTPLUS	Softplus	
AF_SOFTSIGN	Softsign	
AF_SWISH	Função Swish	

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Parâmetros
AF_TANH	Tangente hiperbólica	
AF_TRELU	Retificador linear com limiar	

ENUM_SORT_MODE

Enumeração dos tipos de classificação para a função Sort.

Identificador	Descrição
SORT_ASCENDING	Classificação ascendente
SORT_DESCENDING	Classificação decrescente

ENUM_MATRIX_AXIS

Enumeração para indicar o eixo em todas as funções estatísticas para matrizes.

Identificador	Descrição
AXIS_NONE	O eixo não é definido, o cálculo é realizado em todos os elementos da matriz, como se fosse um vetor (consulte o método Flat).
AXIS_HORZ	Eixo horizontal
AXIS_VERT	Eixo vertical

© 2000-2024, MetaQuotes Ltd.

Inicialização

Existem várias maneiras de declarar e inicializar matrizes e vetores.

Função	Ação
Assign	Copia uma matriz, vetor ou array com conversão automática
CopyIndicatorBuffer	Obtém no vetor dados do buffer especificado do indicador especificado com a quantidade indicada
CopyRates	Obtém em uma matriz ou vetor a série histórica da estrutura MqlRates do período-símbolo especificado na quantidade especificada
CopyTicks	Obtém na matriz ou vetor os ticks a partir da estrutura MqlTick
CopyTicksRange	Obtém na matriz ou vetor os ticks a partir da estrutura MqlTick dentro do intervalo de datas especificado
Eye	Retorna uma matriz com uns na diagonal principal e zeros em todas as outras posições.
Identity	Cria uma matriz de identidade do tamanho especificado
Ones	Cria e retorna uma nova matriz preenchida com uns
Zeros	Cria e retorna uma nova matriz preenchida com zeros
Full	Cria e retorna uma nova matriz preenchida com os valores fornecidos
Tri	Cria uma matriz com uns na diagonal indicada e abaixo e zeros em todas as outras posições
Init	Inicializa uma matriz ou vetor
Fill	Preenche uma matriz ou vetor existente com o valor fornecido

Declaração sem especificar um tamanho (sem alocação de memória de dados):

```
matrix         matrix_a;   // matriz do tipo double
matrix<double> matrix_a1;  // outra forma de declarar uma matriz de tipo double, e e
matrixf        matrix_a2;  // matriz do tipo float
matrix<float>  matrix_a3;  // matriz do tipo float
vector         vector_a;   // vetor do tipo double
vector<double> vector_a1;
vectorf        vector_a2;  // vetor do tipo float
vector<float>  vector_a3;
```

Declaração de tamanho (com alocação de memória para dados, mas sem nenhuma inicialização):

```
matrix         matrix_a(128,128);           // podem ser especificados como parâmetr
matrix<double> matrix_a1(InpRows,InpCols);  // ou variáveis
matrixf        matrix_a2(1,128);            // análogo de um vetor horizontal
```

© 2000-2024, MetaQuotes Ltd.

```
matrix<float>  matrix_a3(InpRows,1);        // análogo de um vetor vertical
vector         vector_a(256);
vector<double> vector_a1(InpSize);
vectorf        vector_a2(SomeFunc());       // a função SomeFunc retorna um número c
vector<float>  vector_a3(InpSize+16);       // a expressão pode ser usada como um pa
```

Declaração com inicialização (os tamanhos das matrizes e vetores neste caso são determinados pela sequência de inicialização):

```
matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
matrix<double> matrix_a1=matrix_a;                     // devem ser matrizes do mes
matrixf        matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
matrix<float>  matrix_a3={{1,2},{3,4}};
vector         vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
vector<double> vector_a1={1,5,2.4,3.3};
vectorf        vector_a2={0,1,2,3};
vector<float>  vector_a3=vector_a2;                    // devem ser vetores do mesm
```

Declaração com inicialização:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); // primeiro é criada uma mat
  matrixf matrix_a1(10,20,MatrixArange);                  // após a criação da matriz,
  vector  vector_a(size,VectorArange,-10.0);              // Após a criação de um veto
  vectorf vector_a1(128,VectorArange);
```

É importante notar que as dimensões de uma matriz ou vetor podem ser alteradas, pois a memória de dados é sempre dinâmica.

© 2000-2024, MetaQuotes Ltd.

Métodos estáticos

Métodos estáticos para criar matrizes e vetores do tamanho especificado, inicializados de uma determinada maneira:

```
matrix         matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf        matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix         matrix_a4=matrix::Tri(4,5,-1);
vector         vector_a =vector::Ones(256);
vectorf        vector_a1=vector<float>::Zeros(16);
vector<float>  vector_a2=vectorf::Full(128,float_value);
```

Métodos para inicializar matrizes e vetores já criados:

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

© 2000-2024, MetaQuotes Ltd.

Assign

Copia uma matriz, vetor ou array com conversão automática.

```
bool matrix::Assign(
  const matrix<T>  &mat     // matriz a ser copiada
   );
bool matrix::Assign(
  const void       &array[] // matriz a ser copiada
   );
bool vector::Assign(
  const vector<T>  &vec     // vetor a ser copiado
   );
bool vector::Assign(
  const void       &array[] // matriz a ser copiada
   );
```

Parâmetros

m, v ou array

[in] Matriz, vetor ou array de onde os valores são copiados.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

Ao invés de Copy, o método Assign permite também copiar arrays. Ao mesmo tempo, é feita automaticamente uma conversão de tipo, e a matriz ou vetor resultante é ajustado ao tamanho do array que está sendo copiado.

Exemplo:

```
//--- cópia da matriz
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//--- cópia do array para a matriz
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix before Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
```

© 2000-2024, MetaQuotes Ltd.

```
double_matrix.Assign(int_arr);
Print("double_matrix after Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix before Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix after Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

Ver também

Copy

© 2000-2024, MetaQuotes Ltd.

CopyIndicatorBuffer

Obtém no vetor dados do buffer especificado do indicador especificado com a quantidade indicada.

Os dados no vetor são copiados de forma que o elemento mais antigo no tempo esteja no início da memória física alocada para o vetor. Existem três alternativas para a função.

Manuseio por posição inicial e número de itens necessários

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // identificador do indicador
   ulong     buffer_index,         // número do buffer do indicador
   ulong     start_pos,            // ponto de partida
   ulong     count                 // quantidade que copiamos
   );
```

Manuseio por data inicial e número de itens necessários

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // identificador do indicador
   ulong     buffer_index,         // número do buffer do indicador
   datetime  start_time,           // data de início
   ulong     count                 // quantidade que copiamos
   );
```

Manuseio por datas de início e término do intervalo de tempo necessário

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // identificador do indicador
   ulong     buffer_index,         // número do buffer do indicador
   datetime  start_time,           // data de início
   datetime  stop_time             // data de término
   );
```

Parâmetros

indicator_handle

[in] Identificador do indicador obtido pela função de indicador apropriada.

buffer_index

[in] Número do buffer de indicador.

start_pos

[in] Número do primeiro item a ser copiado.

count

[in] Número de itens a serem copiados.

start_time

[in] Tempo de barra correspondente ao primeiro item.

stop_time

[in] Tempo de barra correspondente ao último item.

© 2000-2024, MetaQuotes Ltd.

Valor retornado

Retorna true se for bem-sucedido; caso contrário, false se estiver incorreto.

Observação

Os elementos de dados a serem copiados (buffer do indicador com índice buffer_index) são contados a partir da posição inicial do presente para o passado, ou seja, uma posição inicial igual a 0 representa a barra atual (valor do indicador para a barra atual).

Ao copiar antecipadamente uma quantidade desconhecida de dados, é recomendável declarar um vetor sem especificar o tamanho (sem alocar memória para os dados), pois a função CopyBuffer() tenta alocar o tamanho do vetor receptor de acordo com o tamanho dos dados que estão sendo copiados.

Se for necessária uma cópia parcial dos valores do indicador, um vetor intermediário deverá ser usado para essa finalidade, vetor esse que receberá a cópia da quantidade necessária. E já a partir desse vetor intermediário, é feita a cópia item por item do número necessário de valores para os locais corretos do vetor receptor.

Se uma quantidade predefinida de dados tiver que ser copiada, é melhor declarar um vetor com tamanho especificado antecipadamente para evitar a alocação desnecessária de memória.

Ao solicitar dados de um indicador, se a série temporal solicitada ainda não tiver sido criada ou precisar ser baixada do servidor, a função retornará imediatamente false, mas o próprio processo de download/criação será iniciado.

Ao solicitar dados de um Expert Advisor ou script, ele iniciará o carregamento a partir do servidor se o terminal não tiver esses dados localmente ou começará a criar as séries temporais necessárias se os dados puderem ser criados a partir do histórico local, mas ainda não estiverem prontos. A função retornará a quantidade de dados que estarão prontos quando o tempo limite expirar.

Veja também

CopyBuffer

© 2000-2024, MetaQuotes Ltd.

CopyRates

Obtém na matriz ou vetor as séries históricas da estrutura MqlRates do período-símbolo especificado com a quantidade especificada. Os itens da posição inicial são contados do presente para o passado, ou seja, uma posição inicial de 0 significa a barra atual.

Além disso, os dados serão copiados de tal forma que o elemento mais antigo no tempo seja colocado no início da matriz/vetor. Existem 3 variantes do método.

Acesso através da posição inicial e número de itens necessários.

```
bool  matrix::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   ulong            start,         // número da barra inicial a partir da qual começar
   ulong            count          // quantidade a copiar
   );
```

Acesso através da data inicial e número de itens necessários.

```
bool  matrix::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   datetime         from,          // desde que data
   ulong            count          // quantidade a copiar
   );
```

Acesso através das datas inicial e final dentro de um intervalo de tempo.

```
bool  matrix::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   datetime         from,          // desde que data
   datetime         to             // até que data
   );
```

Para um vetor, os métodos são semelhantes.

Acesso através da posição inicial e número de itens necessários.

```
bool  vector::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   ulong            start,         // número da barra inicial a partir da qual começar
   ulong            count          // quantidade a copiar
   );
```

Acesso através da data inicial e número de itens necessários.

© 2000-2024, MetaQuotes Ltd.

```
bool  vector::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   datetime         from,          // desde que data
   ulong            count          // quantidade a copiar
   );
```

Acesso através das datas inicial e final dentro de um intervalo de tempo.

```
bool  vector::CopyRates(
   string           symbol,        // nome do símbolo
   ENUM_TIMEFRAMES  period,        // período
   ulong            rates_mask,    // combinação de sinalizadores para indicar a série
   datetime         from,          // desde que data
   datetime         to             // até que data
   );
```

Parâmetros

symbol

[in] Símbolo.

period

[in] Período.

rates_mask

[in] Combinação de sinalizadores provenientes da enumeração ENUM_RATES que indicam o tipo de série solicitada. Ao copiar para um vetor, apenas um valor da enumeração ENUM_RATES pode ser especificado, caso contrário, ocorrerá um erro.

start

[in] Número do primeiro item copiado.

count

[in] Número de itens a serem copiados.

from

[in] Tempo de barra correspondente ao primeiro item.

to

[in] Tempo de barra correspondente ao último item.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false se for erro.

Observação

Se o intervalo de dados solicitado estiver totalmente fora dos dados disponíveis no servidor, a função retorna false. Se os dados forem solicitados fora de TERMINAL_MAXBARS (número máximo de barras no gráfico), a função também retornará false.

© 2000-2024, MetaQuotes Ltd.

Ao solicitar dados de um EA ou script, será iniciada um carregamento desde o servidor se o terminal não tiver esses dados localmente, ou será iniciada a construção da série temporal necessária se os dados puderem ser construídos a partir do histórico local, mas ainda não estiverem prontos. A função retornará a quantidade de dados que estarão prontos quando o tempo expirar, mas o carregamento do histórico continuará, e na próxima vez que uma solicitação semelhante for feita a função retornará mais dados.

Ao solicitar dados por data de início e número de itens necessários, somente os dados com data menor que (antes) ou igual à data especificada são devolvidos. O intervalo é definido e considerado com uma precisão de segundos. Ou seja, a data de abertura de qualquer barra para a qual um valor é retornado (volume, spread, preço de abertura, máximo, mínimo, fechamento ou hora) é sempre igual ou menor que o especificado.

Ao solicitar dados em um determinado intervalo de datas, apenas os dados que se enquadram no intervalo solicitado são retornados, enquanto o intervalo é definido e considerado com precisão de até um segundo. Ou seja, o tempo de abertura de qualquer barra para a qual um valor é retornado (volume, spread, valor no buffer do indicador, preço de abertura, máximo, mínimo, fechamento ou tempo) está sempre dentro do intervalo solicitado.

Assim, se o dia da semana atual for sábado, quando você tentar copiar dados no período semanal, especificando *start_time=ultima_terça-feira* e *stop_time=ultima_sexta-feira* a função retornará 0, pois o horário de abertura no período semanal sempre cai no domingo, mas nenhuma barra semanal está dentro do intervalo especificado.

Se você precisar obter o valor correspondente à barra incompleta atual, poderá usar a primeira forma da chamada, indicando *start_pos*=0 и *count*=1.

ENUM_COPY_RATES

A enumeração ENUM_COPY_RATES contém sinalizadores para especificar o tipo de dados a ser passado para a matriz ou array. A combinação de sinalizadores permite obter várias séries do histórico em uma solicitação. Nesse caso, a ordem das linhas na matriz corresponderá à ordem dos valores na enumeração ENUM_COPY_RATES, ou seja, a linha com dados High será sempre maior na matriz do que a linha com dados Low.

Identificador	Valor	Descrição
COPY_RATES_OPEN	1	Série de preços Open
COPY_RATES_HIGH	2	Série de preços High
COPY_RATES_LOW	4	Série de preços Low
COPY_RATES_CLOSE	8	Série de preços Close
COPY_RATES_TIME	16	Série Time (hora de abertura da barra) A obtenção de tempo no valor float do vetor e matriz (vectord e matrixf) acarreta a perda de ~100 segundos, em outras palavras a precisão do valor float é fortemente limitado, e inteiros com um valor superior a

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Descrição
		1<<24 não podem ser representados com precisão em um valor float.
COPY_RATES_VOLUME_TICK	32	Volumes de ticks
COPY_RATES_VOLUME_REAL	64	Volumes de negociação
COPY_RATES_SPREAD	128	Spreads
Combinação		
COPY_RATES_OHLC	15	Séries Open, High, Low e Close
COPY_RATES_OHLCT	31	Série Open, Hilgh, Low, Close e Time
Disposição dos dados		
COPY_RATES_VERTICAL	32768	As séries são copiadas na matriz ao longo do eixo vertical. Isso implica que os valores das séries na matriz são organizados de forma vertical, ou seja, os dados mais antigos são apresentados na primeira linha e os mais recentes, na última. Por padrão, ao copiar para uma matriz, as séries são adicionadas ao longo do eixo horizontal. O sinalizador só é relevante ao copiar para uma matriz.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- obtemos as cotações na matriz
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("matrix_rates.CopyRates failed. Error ", GetLastError());
//--- verificação
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
    ArrayPrint(mql_rates);
   }
  else
```

© 2000-2024, MetaQuotes Ltd.

```
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ", GetLastErro
//--- obtemos cotações no vetor = chamada errada
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Error ", GetLastError());
//--- obtemos os preços de fechamento no vetor
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("vector_rates.CopyRates failed. Error ", GetLastError());
 };
/*
   matrix rates:
   [[0.99686,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.995280
    [0.99708,0.99643,0.99591,0.9955000000000001,0.99652,0.99795,0.99865,0.99764,0.996
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,166
   mql_rates array:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641          4463        0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588          4519        0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441          3060        0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464          3867        0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594          5280        0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697          7227        0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758         10130        0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581          7012        0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528          6166        0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259          6950        0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Error 4003
   vector_rates COPY_RATES_CLOSE:
   [0.9931,0.99293,0.99417,0.99504,0.9968399999999999,0.99641,0.99588,0.99441,0.99464,
*/
```

Ver também

Acesso a séries temporais e indicadores, CopyRates

© 2000-2024, MetaQuotes Ltd.

CopyTicks

Obtém em uma matriz ou vetor os ticks a partir da estrutura MqlTick. Os elementos da posição inicial são contados do passado para o presente, ou seja, o tick com índice 0 é o mais antigo. Para analisar um tick é necessário verificar o campo *flags*, que notifica exatamente o que foi alterado no tick em questão.

```
bool  matrix::CopyTicks(
   string          symbol,                // nome do símbolo
   ulong           ticks_mask,            // máscara que define o tipo de ticks recek
   uint            flags=COPY_TICKS_ALL,  // sinalizador que define o tipo de ticks r
   ulong           from_msc=0,            // momento a partir do qual os ticks são sc
   ulong           count=0                // número de ticks a serem obtidos
   );
```

Método vetorial

```
bool  vector::CopyTicks(
   string          symbol,                // nome do símbolo
   ulong           ticks_mask,            // máscara que define o tipo de ticks recek
   uint            flags=COPY_TICKS_ALL,  // sinalizador que define o tipo de ticks r
   ulong           from_msc=0,            // momento a partir do qual os ticks são sc
   ulong           count=0                // número de ticks a serem obtidos
   );
```

Parâmetros

symbol

[in] Símbolo.

ticks_mask

[in] Combinação de sinalizadores a partir da enumeração ENUM_COPY_TICKS, que indica a composição dos dados solicitados. Ao copiar em um vetor, apenas um valor da enumeração pode ser especificado a partir da enumeração ENUM_COPY_TICKS, caso contrário, haverá um erro.

flags

[in] Sinalizador que define o tipo de tick solicitado. COPY_TICKS_INFO - ticks causados por mudanças em Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações Last e Volume, COPY_TICKS_ALL - todos os ticks. Após qualquer tipo de solicitação, os valores do tick anterior são adicionados aos demais campos da estrutura MqlTick.

from_msc

[in] momento a partir do qual os ticks são solicitados. Indicado em milissegundos a partir de 01.01.1970. Se o parâmetro from_msc=0, então é dada a última count de ticks.

count

[in] Número de ticks solicitados. Se os parâmetros from_msc e count não forem especificados, todos os últimos ticks disponíveis serão registrados, mas não mais do que 2000.

© 2000-2024, MetaQuotes Ltd.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false se for erro.

Observação

A primeira chamada a CopyTicks() inicia uma sincronização do banco de dados de ticks armazenados no disco rígido para um determinado símbolo. Se não houver ticks suficientes no banco de dados local, os ticks ausentes serão baixados automaticamente do servidor de negociação. Neste caso, os ticks são sincronizados a partir do momento ***from_msc***, especificado em CopyTicks(), até o momento. Após essa operação, todos os ticks recebidos para esse símbolo serão armazenados no banco de dados de ticks, mantendo-o atualizado em tempo real e em sincronia.

Se os parâmetros *from_msc* e *count* não forem especificados, então todos os ticks disponíveis serão escritos para a matriz/vetor, mas não mais do que 2000.

Nos indicadores, o método CopyTicks() retorna o resultado imediatamente: Quando chamado a partir do indicador CopyTick() retornará imediatamente os ticks disponíveis por símbolo, e também iniciará a sincronização do banco de dados de ticks se houver dados insuficientes. Todos os indicadores de um mesmo símbolo operam em um fluxo comum, portanto, um indicador não deve aguardar a conclusão da sincronização para continuar funcionando. Após o término da sincronização, a próxima chamada a CopyTicks() retornará todos os ticks solicitados. A função OnCalculate() nos indicadores é chamada após a recepção de cada tick.

Em EAs e scripts, o método CopyTicks() pode esperar até 45 segundos por um resultado: Ao contrário de um indicador, cada EA e script trabalha em seu próprio fluxo e pode, portanto, esperar até 45 segundos para que a sincronização seja concluída. Se durante esse tempo os ticks não forem sincronizados na medida necessária, então CopyTicks() retornará apenas os ticks disponíveis por timeout, enquanto a sincronização continuará. A função OnTick() nos EAs não é um manipulador de cada tick, ela apenas notifica o EA sobre as mudanças no mercado. As mudanças no mercado podem ocorrer em lotes, ou seja, diversos ticks podem ser recebidos pelo terminal simultaneamente. Entretanto, a função OnTick() será chamada apenas uma vez para notificar o Expert Advisor sobre a última condição do mercado.

Taxa de saída: o terminal mantém em cache os últimos 4096 ticks de cada símbolo (65536 ticks para símbolos com a pilha em execução) para acesso rápido, o que permite consultas mais rápidas aos dados históricos. Quando são solicitados ticks da sessão de negociação atual que não estão armazenados em cache, a função CopyTicks() acessa diretamente os ticks armazenados na memória do terminal. No entanto, essas solicitações podem levar mais tempo para serem executadas. As solicitações de ticks de outros dias são as mais lentas, pois nesse caso os dados são lidos do disco, o que leva mais tempo para serem recuperados e processados.

ENUM_COPY_TICKS

A enumeração ENUM_COPY_TICKS contém sinalizadores para especificar o tipo de dados a ser passado para a matriz ou array. A combinação de sinalizadores permite obter várias séries do histórico em uma solicitação. Nesse caso, a ordem das linhas na matriz corresponderá à ordem dos valores na enumeração ENUM_COPY_TICKS, ou seja, a linha com dados High será sempre maior na matriz do que a linha com dados Low.

Identificador	Valor	Descrição
COPY_TICKS_TIME_MS	1	Tempo de tick em milissegundos

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Descrição
COPY_TICKS_BID	2	Preço Bid
COPY_TICKS_ASK	4	Preço Ask
COPY_TICKS_LAST	8	Preço Last (preço da última negociação)
COPY_TICKS_VOLUME	16	Volume para o preço Last
COPY_TICKS_FLAGS	32	Sinalizador de ticks
Disposição dos dados		
COPY_TICKS_VERTICAL	32768	Os ticks são copiados na matriz ao longo do eixo vertical. Isso implica que os ticks na matriz são organizados de forma vertical, ou seja, os ticks mais antigos são apresentados na primeira linha e os mais recentes, na última. Por padrão, ao copiar para uma matriz, os ticks são adicionadas ao longo do eixo horizontal. O sinalizador só é relevante ao copiar para uma matriz.

Para saber exatamente quais dados foram alterados com o tick atual, analise seus sinalizadores:

- TICK_FLAG_BID - tick mudou o preço Bid
- TICK_FLAG_ASK - tick mudou o preço Ask
- TICK_FLAG_LAST - tick mudou o preço da última negociação
- TICK_FLAG_VOLUME - tick modou o volume
- TICK_FLAG_BUY - tick surgiu como resultado de uma negociação de compra
- TICK_FLAG_SELL - tick surgiu como resultado de uma negociação de venda

Ver também

Acesso a séries temporais e indicadores, CopyTicks

© 2000-2024, MetaQuotes Ltd.

CopyTicksRange

Obtém em uma matriz ou vetor os ticks a partir da estrutura MqlTick na faixa de datas indicada. Os elementos da posição inicial são contados do passado para o presente, ou seja, o tick com índice 0 é o mais antigo. Para analisar um tick é necessário verificar o campo *flags*, que notifica exatamente o que foi alterado no tick em questão.

```
bool  matrix::CopyTicksRange(
   string          symbol,                // nome do símbolo
   ulong           ticks_mask,            // máscara que define o tipo de ticks recek
   uint            flags=COPY_TICKS_ALL,  // sinalizador que define o tipo de ticks r
   ulong           from_msc=0,            // momento a partir do qual os ticks são sc
   ulong           to_msc=0               // momento a partir do qual os ticks são sc
   );
```

Método vetorial

```
bool  vector::CopyTicksRange(
   string          symbol,                // nome do símbolo
   ulong           ticks_mask,            // máscara que define o tipo de ticks recek
   uint            flags=COPY_TICKS_ALL,  // sinalizador que define o tipo de ticks r
   ulong           from_msc=0,            // momento a partir do qual os ticks são sc
   ulong           to_msc=0               // momento a partir do qual os ticks são sc
   );
```

Parâmetros

symbol

[in] Símbolo.

ticks_mask

[in] Combinação de sinalizadores a partir da enumeração ENUM_COPY_TICKS, que indica a composição dos dados solicitados. Ao copiar em um vetor, apenas um valor da enumeração pode ser especificado a partir da enumeração ENUM_COPY_TICKS, caso contrário, haverá um erro.

flags

[in] Sinalizador que define o tipo de tick solicitado. COPY_TICKS_INFO - ticks causados por mudanças em Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações Last e Volume, COPY_TICKS_ALL - todos os ticks. Após qualquer tipo de solicitação, os valores do tick anterior são adicionados aos demais campos da estrutura MqlTick.

from_msc

[in] momento a partir do qual os ticks são solicitados. Indicado em milissegundos a partir de 01.01.1970. Se o parâmetro from_msc não for especificado, os ticks serão fornecidos desde o início do histórico. São entregados ticks com a data >= from_msc.

to_msc

[in] Momento a partir do qual são solicitados os ticks. Indicado em milissegundos a partir de 01.01.1970. São entregados ticks com a data <= to_msc. Se o parâmetro to_msc não é especificado, então todos os ticks são dados até o final do histórico.

Valor retornado

© 2000-2024, MetaQuotes Ltd.

Retorna true se bem sucedido, caso contrário false. GetLastError() pode retornar os seguintes erros:

- ERR_HISTORY_TIMEOUT – o tempo de espera para a sincronização dos ticks expirou, a função entregou tudo o que tinha.
- ERR_HISTORY_SMALL_BUFFER – o buffer estático é muito pequeno, apenas entregou o que cabia na matriz.
- ERR_NOT_ENOUGH_MEMORY – memória insuficiente para obter o histórico do intervalo especificado em uma matriz dinâmica de ticks. Falha ao alocar a quantidade necessária de memória para a matriz de ticks.

Para saber exatamente quais dados foram alterados com o tick atual, analise seus sinalizadores:

- TICK_FLAG_BID – tick mudou o preço Bid
- TICK_FLAG_ASK – tick mudou o preço Ask
- TICK_FLAG_LAST – tick mudou o preço da última negociação
- TICK_FLAG_VOLUME – tick modou o volume
- TICK_FLAG_BUY – tick surgiu como resultado de uma negociação de compra
- TICK_FLAG_SELL – tick surgiu como resultado de uma negociação de venda

Observação

O método CopyTicksRange() é projetado para solicitar ticks de uma faixa estritamente especificada, por exemplo, para um dia em particular do histórico. Enquanto CopyTicks() permite especificar apenas uma data de início, como por exemplo receber todos os ticks desde o início do mês até a hora atual.

Ver também

Acesso a séries temporais e indicadores, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

Eye

A função estática cria uma matriz do tamanho especificado com uns na diagonal especificada e zeros fora da diagonal. Retorna uma matriz com uns ao longo da diagonal e zeros em todas as outras posições.

```
static matrix matrix::Eye(
  const ulong  rows,        // número de filas
  const ulong  cols,        // número de colunas
  const int    ndiag=0      // índice diagonal
   );
```

Parâmetros

rows

[in] Número de linhas de saída.

cols

[in] Número de colunas de saída.

ndiag=0

[in] Índice da diagonal: 0 (padrão) representa a diagonal principal, um valor positivo representa uma diagonal superior e um valor negativo representa uma diagonal inferior.

Valor retornado

Matriz na qual todos os elementos são iguais a zero, exceto a k-ésima diagonal, cujos valores são iguais a um.

Exemplo em MQL5:

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

Exemplo em Python:

```
np.eye(3, dtype=int)
```

© 2000-2024, MetaQuotes Ltd.

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])

 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

© 2000-2024, MetaQuotes Ltd.

Identity

A função estática cria uma matriz identidade do tamanho especificado (não necessariamente quadrada). A matriz identidade contém uns na diagonal principal e zeros fora da diagonal principal. A diagonal principal consiste nos elementos de uma matriz cujos índices de linha e coluna são iguais, ou seja, [0,0],[1,1],[2,2], etc. Cria uma nova matriz de identidade.

Existe também um método Identity que converte uma matriz já existente em uma identidade.

```
static matrix matrix::Identity(
  const ulong  rows,         // número de filas
  const ulong  cols,         // número de colunas
   );

void matrix::Identity();
```

Parâmetros

rows

[in] Número de linhas (e colunas) em uma matriz n x n.

Valor retornado

Retorna a matriz identidade. A matriz identidade é uma matriz quadrada com uns na diagonal principal.

Exemplo em MQL5:

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

Exemplo em Python:

```
np.identity(3)
```

© 2000-2024, MetaQuotes Ltd.

```
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
       [0.,  0.,  1.]])
```

© 2000-2024, MetaQuotes Ltd.

Ones

Função estática. Cria e retorna uma nova matriz preenchida com uns.

```
static matrix matrix::Ones(
  const ulong  rows,      // número de linhas
  const ulong  cols       // número de colunas
   );

static vector vector::Ones(
  const ulong  size,      // tamanho do vetor
   );
```

Parâmetros

rows

[in] Número de filas.

cols

[in] Número de colunas.

Valor retornado

Retorna uma nova matriz a partir das linhas e colunas dadas, matriz essa preenchida com uns.

Exemplo em MQL5:

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

Exemplo em Python:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2024, MetaQuotes Ltd.

Zeros

Função estática. Cria e retorna uma nova matriz preenchida com zeros.

```
static matrix matrix::Zeros(
  const ulong  rows,      // número de linhas
  const ulong  cols       // número de colunas
   );

static vector vector::Zeros(
  const ulong  size,      // tamanho do vetor
   );
```

Parâmetros

rows

[in] Número de filas.

cols

[in] Número de colunas.

Valor retornado

Retorna uma nova matriz a partir das linhas e colunas dadas, matriz essa preenchida com zeros.

Exemplo em MQL5:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Exemplo em Python:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000-2024, MetaQuotes Ltd.

Full

Função estática. Cria e retorna uma nova matriz preenchida com o valor especificado.

```
static matrix matrix::Full(
  const ulong   rows,       // número de linhas
  const ulong   cols,       // número de colunas
  const double  value       // valor para preenchimento
   );

static vector vector::Full(
  const ulong   size,       // tamanho do vetor
  const double  value       // valor para preenchimento
   );
```

Parâmetros

rows

[in] Número de filas.

cols

[in] Número de colunas.

value

[in] Valor com o qual todos os elementos da matriz serão preenchidos.

Valor retornado

Retorna uma nova matriz a partir das linhas e colunas fornecidas, matriz essa preenchida com o valor especificado.

Exemplo em MQL5:

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

Exemplo:

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000-2024, MetaQuotes Ltd.

Tri

Função estática. Cria uma matriz com uns na diagonal indicada e abaixo e zeros em todas as outras posições.

```
static matrix matrix::Tri(
  const ulong rows,         // número de linhas
  const ulong cols,         // número de colunas
  const int   ndiag=0       // número de diagonais
   );
```

Parâmetros

rows

[in] Número de linhas na matriz.

cols

[in] Número de colunas na matriz.

ndiag=0

[in] A subdiagonal da matriz que, junto com as demais abaixo dela, será preenchida. k = 0 — diagonal principal, k < 0 — diagonais abaixo da diagonal principal, k > 0 — acima. Por padrão é 0.

Valor retornado

Uma matriz cujo triângulo inferior é preenchido com uns e todos os demais elementos com zeros.

Exemplo em MQL5:

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

Exemplo:

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

© 2000-2024, MetaQuotes Ltd.

Init

Inicializa uma matriz ou vetor.

```
void matrix::Init(
  const ulong  rows,            // número de linhas
  const ulong  cols,            // número de colunas
   func_name   init_func=NULL,  // função de inicialização colocada em algum área de v
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // tamanho do vetor
  func_name    init_func=NULL,  // função de inicialização colocada em alguma área de
   ...         parameters
   );
```

Parâmetros

rows

[in] Número de filas.

cols

[in] Número de colunas.

func_name

[in] Função de inicialização.

...

[in] Parâmetros da função de inicialização.

Valor retornado

Não há valor de retorno.

Exemplo:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // primeiro cria uma matriz não-inici
  Print("matrix m \n",m);                        // em seguida, a função MatrixArange
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // após a criação de uma matriz do ti
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // após a criação de um vetor, Vector
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.999999999999992,-0.899999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

© 2000-2024, MetaQuotes Ltd.

Fill

Preenche uma matriz ou vetor existente com o valor fornecido.

```
void matrix::Fill(
  const double  value      // valor para preenchimento
   );

void vector::Fill(
  const double  value      // valor para preenchimento
   );
```

Parâmetros

value

[in] Valor com o qual todos os elementos da matriz serão preenchidos.

Valor retornado

Não há valor de retorno. A matriz é preenchida com o valor especificado localmente.

Exemplo:

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Manuseio de matrizes e vetores.

Métodos para manipulação básica de matrizes, incluindo preenchimento, cópia, obtenção de submatrizes, transposição, divisão e classificação.

Existem também vários métodos para trabalhar com as linhas e colunas de uma matriz.

Função	Ação
HasNan	Retorna o número de valores NaN na matriz/vetor
Transpose	Expande ou reorganiza os eixos de uma matriz, retorna a matriz modificada
TriL	Retorna uma cópia da matriz, substituindo todos os elementos acima da k-ésima diagonal por zeros. Matriz triangular inferior
TriU	Retorna uma cópia da matriz, substituindo todos os elementos abaixo da k-ésima diagonal por zeros. Matriz triangular superior
Diag	Extrai uma diagonal ou cria uma matriz diagonal
Row	Retorna um vetor linha. Grava um vetor na string especificada
Col	Retorna um vetor coluna. Grava um vetor na coluna especificada
Copy	Devolve uma cópia da matriz/vetor em questão
Compare	Compara os elementos de duas matrizes/vetores com a precisão especificada
CompareByDigits	Compara se os elementos de duas matrizes/vetores correspondem, levando em conta dígitos significativos
Flat	Permite endereçar um elemento da matriz por meio de um índice, e não por meio de dois
Clip	Limita os elementos de uma matriz/vetor a um intervalo especificado de valores válidos
Reshape	Altera a forma de uma matriz sem mudar seus dados
Resize	Retorna uma nova matriz da forma e tamanho especificados
SwapRows	Troca linhas em uma matriz
SwapCols	Troca colunas em uma matriz
Split	Divisão de uma matriz em várias submatrices
Hsplit	Divisão horizontal de uma matriz em várias submatrizes. O mesmo que Split c axis=0
Vsplit	Divisão vertical de uma matriz em várias submatrizes. O mesmo que Split c axis=1
ArgSort	Executa uma classificação indireta em uma matriz ou vetor.
Sort	Classifica no local uma matriz ou vetor.

© 2000-2024, MetaQuotes Ltd.

HasNan

Retorna o número de valores NaN na matriz/vetor.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Valor retornado

Número de elementos de matriz/vetor que contêm um valor NaN.

Observação

Os métodos Compare e CompareByDigits, ao comparar um par de elementos correspondentes com valores NaN, consideram esses elementos iguais, enquanto em uma comparação normal de ponto flutuante NaN != NaN.

Exemplo:

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vector: ", v1.Compare(v2,0)==0);
  }

/* Resultado:

 single: false
 vector: true
*/
```

Veja também

MathClassify, Compare, CompareByDigits

© 2000-2024, MetaQuotes Ltd.

Transpose

Transposição de matrizes. Expande ou reorganiza os eixos de uma matriz, retorna a matriz modificada.

```
matrix matrix::Transpose()
```

Valor retornado

Matriz transposta.

Algoritmo simples da transposição de matriz em MQL5:

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

Exemplo em MQL5:

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

Exemplo em Python:

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000-2024, MetaQuotes Ltd.

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000-2024, MetaQuotes Ltd.

TriL

Retorna uma cópia da matriz, substituindo todos os elementos acima da k-ésima diagonal por zeros. Matriz triangular inferior.

```
matrix matrix::Tril(
  const int     ndiag=0      // índice diagonal
   );
```

Parâmetros

ndiag=0

[in] A diagonal acima da qual os elementos são substituídos por zeros. ndiag = 0 (por padrão) — diagonal principal, ndiag < 0 — abaixo, já ndiag > 0 — acima da diagonal principal.

Valor retornado

Uma matriz cujo triângulo inferior é preenchido com uns e todos os demais elementos com zeros.

Exemplo em MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

Exemplo em Python:

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

© 2000-2024, MetaQuotes Ltd.

TriU

Retorna uma cópia da matriz, substituindo todos os elementos abaixo da k-ésima diagonal por zeros. Matriz triangular superior.

```
matrix matrix::Triu(
  const int     ndiag=0      // índice diagonal
   );
```

Parâmetros

ndiag=0

[in] Diagonal abaixo da qual os elementos são substituídos por zeros. ndiag = 0 (por padrão) — diagonal principal, ndiag < 0 — abaixo, já ndiag > 0 — acima da diagonal principal.

Exemplo em MQL5:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

Exemplo em Python:

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

© 2000-2024, MetaQuotes Ltd.

Diag

Extrai a diagonal ou cria uma matriz diagonal.

```
vector matrix::Diag(
  const int     ndiag=0      // número diagonal
   );

void matrix::Diag(
  const vector  v,           // vetor diagonal
  const int     ndiag=0      // número diagonal
   );
```

Parâmetros

v

[in] Vetor cujos elementos devem ser inseridos na diagonal correspondente (ndiag=0 — diagonal principal).

ndiag=0

[in] Diagonal. Por padrão é 0. Para diagonais acima da diagonal principal, é especificado ndiag>0, para diagonais abaixo é ndiag<0.

Observação

A diagonal pode ser definida para matrizes não distribuídas (não dimensionadas). Neste caso, será criada uma matriz nula do tamanho do vetor diagonal correspondente, após a qual os valores dos elementos vetoriais serão inseridos na diagonal correspondente. Se uma diagonal for definida em uma matriz existente, as dimensões da matriz permanecerão as mesmas e os valores dos elementos da matriz fora do vetor diagonal não serão alterados.

Exemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Diag(v1);
   Print("m1\n",m1);
   matrix m2;
   m2.Diag(v1,-1);
   Print("m2\n",m2);
   matrix m3;
   m3.Diag(v1,1);
   Print("m3\n",m3);
   matrix m4=matrix::Full(4,5,9);
   m4.Diag(v1,1);
   Print("m4\n",m4);

   Print("diag -1 - ",m4.Diag(-1));
```

© 2000-2024, MetaQuotes Ltd.

```
 Print("diag 0 - ",m4.Diag());
 Print("diag 1 - ",m4.Diag(1));

/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

© 2000-2024, MetaQuotes Ltd.

Row

Retorna um vetor linha. Grava um vetor na string especificada.

```
vector matrix::Row(
  const ulong   nrow      // número da linha
   );

void matrix::Row(
  const vector  v,        // vetor da linha
  const ulong   nrow      // número da linha
   );
```

Parâmetros

nrow

[in] Número da linha.

Valor retornado

Vetor.

Observação

A linha pode ser definida para matrizes não distribuídas (sem dimensões definidas). Nesse caso, será criada uma matriz com todos os elementos iguais a zero, com tamanho (número de linhas + 1) x tamanho do vetor, após o que os valores dos elementos do vetor serão inseridos na linha correspondente. Se a linha for definida para uma matriz já existente, as dimensões da matriz não serão alteradas e os valores dos elementos da matriz fora do vetor da linha não serão alterados.

Exemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
  [[7,7,7,7,7]
```

© 2000-2024, MetaQuotes Ltd.

```
[7,7,7,7,7]
[1,2,3,7,7]
[7,7,7,7,7]]
row 1 - [7,7,7,7,7]
row 2 - [1,2,3,7,7]
*/
```

© 2000-2024, MetaQuotes Ltd.

Col

Retorna um vetor coluna. Grava um vetor na coluna especificada.

```
vector matrix::Col(
  const ulong   ncol      // número da coluna
   );

void matrix::Col(
  const vector  v,        // vetor da coluna
  const ulong   ncol      // número da coluna
   );
```

Parâmetros

ncol

[in] Número da coluna.

Valor retornado

Vetor.

Observação

Uma coluna pode ser definida para matrizes não resolvidas (sem dimensões). Nesse caso, será criada uma matriz nula com tamanho tamanho do vetor x (número de colunas + 1), após o que os valores dos elementos do vetor serão inseridos na coluna correspondente. Se a coluna for definida para uma matriz já existente, as dimensões da matriz não serão alteradas e os valores dos elementos da matriz fora do vetor da coluna não serão alterados.

Exemplo:

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
  m2
  [[8,8,1,8,8]
```

© 2000-2024, MetaQuotes Ltd.

```
[8,8,2,8,8]
[8,8,3,8,8]
[8,8,8,8,8]]
col 1 - [8,8,8,8]
col 2 - [1,2,3,8]
*/
```

© 2000-2024, MetaQuotes Ltd.

Copy

Cria uma cópia da matriz/vetor fornecido.

```
bool matrix::Copy(
  const matrix&  a     // matriz de cópia
   );
bool vector::Copy(
  const vector&  v     // vetor de cópia
   );
```

Parâmetros

v

[in] Matriz ou vetor a ser copiado.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo em MQL5:

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

Exemplo em Python:

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2024, MetaQuotes Ltd.

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

Ver também

Assign

© 2000-2024, MetaQuotes Ltd.

Compare

Compara os elementos de duas matrizes/vetores com a precisão especificada.

```
ulong vector::Compare(
  const vector& vec,          // vetor de comparação
  const double  epsilon       // precisão
   );

ulong matrix::Compare(
  const matrix& mat,          // matriz de comparação
  const double  epsilon       // precisão
   );
```

Parâmetros

vector_b

[in] Vetor de comparação.

vector_b

[in] Vetor de comparação.

epsilon

[in] Precisão.

Valor retornado

Retorna o número de elementos diferentes entre as matrizes ou vetores comparados, ou seja, 0 se as matrizes forem iguais, caso contrário, o valor será maior que 0.

Observação

Os operadores de comparação == ou != produzem uma comparação exata elemento por elemento. É conhecido que a comparação exata de números reais tem sua aplicação limitada, por isso foi adicionado o método de comparação com epsilon. Existem casos em que dentro da mesma matriz existem elementos no intervalo, por exemplo, de 1e-20 a 1e+20. Para isso, é fornecida uma comparação elemento a elemento, levando em consideração dígitos significativos.

Exemplo:

```
matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
matrix matrix_i=matrix::Identity(3,3);
matrix matrix_c=matrix_a.Inv();
matrix matrix_check=matrix_a.MatMul(matrix_c);
Print("matrix_check\n",matrix_check);

ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
```

© 2000-2024, MetaQuotes Ltd.

```
 Print("errors=",errors);


/*
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
errors=0

*/
```

© 2000-2024, MetaQuotes Ltd.

CompareByDigits

Compara os elementos de duas matrizes/vetores para verificar se correspondem com uma precisão de dígitos significativos.

```
ulong vector::CompareByDigits(
  const vector&  vec,          // vetor de comparação
  const int     digits         // número de dígitos significativos
   );

ulong matrix::CompareByDigits(
  const matrix&  mat,          // matriz de comparação
  const int      digits        // número de dígitos significativos
   );
```

Parâmetros

vector_b

[in] Vetor de comparação.

digits

[in] Número de dígitos significativos para comparação.

epsilon

[in] Precisão da comparação. Se a diferença absoluta entre dois valores for menor do que a precisão especificada, eles serão considerados iguais.

Valor retornado

Retorna o número de elementos diferentes entre as matrizes ou vetores comparados, ou seja, 0 se as matrizes forem iguais, caso contrário, o valor será maior que 0.

Observação

Os operadores de comparação == ou != produzem uma comparação exata elemento por elemento. É conhecido que a comparação exata de números reais tem sua aplicação limitada, por isso foi adicionado o método de comparação com epsilon. Existem casos em que dentro da mesma matriz existem elementos no intervalo, por exemplo, de 1e-20 a 1e+20. Para isso, é fornecida uma comparação elemento a elemento, levando em consideração dígitos significativos.

Exemplo:

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- preenchemos a matriz
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      for(int j=0; j<size_k; j++)
        {
         if(i==j)
            matrix_a[i][j]=1.0+i;
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- obtemos outra matriz
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare matrix %d x %d  errors_epsilon=%I64u  errors_digits=%I64u",size_m,s


  /*
  Compare matrix 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

© 2000-2024, MetaQuotes Ltd.

Flat

Permite endereçar um elemento da matriz por meio de um índice, e não por meio de dois.

```
bool matrix::Flat(
  const ulong   index,      //
  const double   value      // valor a ser definido
   );

double matrix::Flat(
  const ulong   index,      //
   );
```

Parâmetros

index

[in] índice Flat

value

[in] O valor a ser definido pelo índice definido.

Valor retornado

Valor com base no índice fornecido.

Observação

Para a matriz matrix mat(3,3) os acessos podem ser escritos da seguinte forma:

- para leitura — x=mat.Flat(4), o que equivale a escrever x=mat[1][1]
- para escrita — mat.Flat(5, 42), o que equivale a escrever mat[1][2]=42

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

© 2000-2024, MetaQuotes Ltd.

```
max_value=12.0
max_value=11.0
*/
```

© 2000-2024, MetaQuotes Ltd.

Clip

Limita os elementos de uma matriz/vetor a um intervalo especificado de valores válidos.

```
bool matrix::Clip(
  const double  min_value,      // valor mínimo
  const double  max_value       // valor máximo
   );
bool vector::Clip(
  const double  min_value,      // valor mínimo
  const double  max_value       // valor máximo
   );
```

Parâmetros

min_value

[in] Valor mínimo.

max_value

[in] Valor máximo.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

A matriz (ou vetor) é processada localmente. Nenhuma cópia é criada.

Exemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

© 2000-2024, MetaQuotes Ltd.

Reshape

Altera a forma de uma matriz sem mudar seus dados.

```
void  Reshape(
  const ulong  rows,      // novo número de linhas
  const ulong  cols       // novo número de colunas
   );
```

Parâmetros

rows

[in] Novo número de linhas.

cols

[in] Novo número de colunas.

Observação

A matriz é processada localmente. Nenhuma cópia é criada. Qualquer dimensão pode ser definida, ou seja rows_new*cols_new!=rows_old*cols_old. Quando o buffer da matriz é aumentado, os valores "extras" ficam indefinidos ou não inicializados.

Exemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
```

© 2000-2024, MetaQuotes Ltd.

```
   [5,6,7,8]]
  */
```

© 2000-2024, MetaQuotes Ltd.

Resize

Retorna uma nova matriz da forma e tamanho especificados.

```
bool matrix::Resize(
  const ulong  rows,      //
  const ulong  cols,      // novo número de colunas
  const ulong  reserve=0 // número reservado de elementos
   );

bool vector::Resize(
  const ulong  size,      // novo tamanho
  const ulong  reserve=0 // número reservado de elementos
   );
```

Parâmetros

rows

[in] Novo número de linhas.

cols

[in] Novo número de colunas.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

A matriz (ou vetor) é processada localmente. Nenhuma cópia é criada. Qualquer dimensão pode ser definida, ou seja rows_new*cols_new!=rows_old*cols_old. Ao contrário do Reshape, a matriz é processada linha por linha. Se o número de colunas aumenta, os valores das colunas "extras" são indefinidos. Quando o número de filas é aumentado, os valores dos elementos das novas filas ficam indefinidos. Quando o número de colunas é reduzido, cada linha da matriz é truncada.

Exemplo:

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);
   matrix_a.Resize(2,6);
   Print("Ressize(2,6)\n",matrix_a);
   matrix_a.Resize(3,5);
   Print("Resize(3,5)\n",matrix_a);
   matrix_a.Resize(2,4);
   Print("Resize(2,4)\n",matrix_a);

   /*
   matrix_a
   [[1,2,3]
```

© 2000-2024, MetaQuotes Ltd.

```
  [4,5,6]
  [7,8,9]
  [10,11,12]]
Ressize(2,6)
[[1,2,3,4,5,6]
 [4,5,6,10,11,12]]
Resize(3,5)
[[1,2,3,4,5]
 [4,5,6,10,11]
 [11,12,3,8,8]]
Resize(2,4)
[[1,2,3,4]
 [4,5,6,10]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Set

Define o valor de um elemento vetorial pelo índice especificado.

```
bool vector::Set(
  ulong   index,      // índice do elemento
  double  value       // valor
   );
```

Parâmetros

index

[in] Índice do elemento para o qual o valor deve ser definido.

value

[in] Valor.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false.

Observação

O método Set faz a mesma coisa que a atribuição de um valor por meio de colchetes, assim: *vector[index]=value*. Adicionado para facilitar a portabilidade do código proveniente de linguagens que usam exatamente essa tipo de entrada. O exemplo abaixo mostra as duas variantes de preenchimento do vetor com valores pelo índice especificado.

Exemplo:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Resultado
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| Preenche o vetor com a potência de um número por meio da operação de atribuição
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

© 2000-2024, MetaQuotes Ltd.

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------+
//| Preenche o vetor com potências de um número usando o método Set
//+------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
     {
      v.Set(k, value);
      value*=2;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SwapRows

Troca as linhas na matriz.

```
bool matrix::SwapRows(
  const ulong  row1,      // índice de primeira linha
  const ulong  row2       // índice da segunda linha
   );
```

Parâmetros

row1

[in] Índice de primeira linha.

row2

[in] Índice da segunda linha.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

© 2000-2024, MetaQuotes Ltd.

SwapCols

Troca colunas em uma matriz.

```
bool matrix::SwapCols(
  const ulong  col1,      // índice da primeira coluna
  const ulong  col2       // índice da segunda coluna
   );
```

Parâmetros

col1

[in] Índice da primeira coluna.

col2

[in] Índice da segunda coluna.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

© 2000-2024, MetaQuotes Ltd.

Split

Divisão de uma matriz em várias submatrices.

```
bool matrix::Split(
  const ulong  parts,      // número de submatrizes
  const int    axis,       // eixo
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );

void matrix::Split(
  const ulong& parts[],    // tamanhos das submatrizes
  const int    axis,       // eixo
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );
```

Parâmetros

parts

[in] Número de submatrices para divisão da matriz.

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

splitted

[out] Matriz de submatrices recebidas.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

Se o número de submatrizes for especificado, serão obtidas submatrizes do mesmo tamanho. Em outras palavras, o tamanho da matriz (0 - número de linhas, 1 - número de colunas) deve ser divisível pelo número parts sem resto. Submatrizes de diferentes tamanhos podem ser obtidas usando uma matriz de tamanhos de submatrizes. Os elementos da matriz de tamanho são usados até que toda a matriz seja dividida. Se a matriz foi completamente dividida e o tamanho restante não for suficiente para formar uma submatriz completa, o restante não dividido será a última submatriz.

Exemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
```

© 2000-2024, MetaQuotes Ltd.

```
 ulong  parts[]={2,2};

 bool res=matrix_a.Split(2,0,splitted);
 Print(res,"  ",GetLastError());
 ResetLastError();
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(2,1,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(parts,0,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

/*
false  4003
true  0
splitted 0
[[1,2,3]
 [7,8,9]
 [13,14,15]
 [19,20,21]
 [25,26,27]]
splitted 1
[[4,5,6]
 [10,11,12]
 [16,17,18]
 [22,23,24]
 [28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
 [7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
 [19,20,21,22,23,24]]
splitted 2
[[25,26,27,28,29,30]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Hsplit

Divisão horizontal de uma matriz em várias submatrizes. O mesmo que Split com axis=0

```
bool matrix::Hsplit(
  const ulong  parts,      // número de submatrizes
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );

void matrix::Hsplit(
  const ulong& parts[],    // tamanhos das submatrizes
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );
```

Parâmetros

parts

[in] Número de submatrices para divisão da matriz.

splitted

[out] Matriz de submatrices recebidas.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

Se o número de submatrizes for especificado, serão obtidas submatrizes do mesmo tamanho. Em outras palavras, o número de linhas deve ser divisível pelo número de partes sem deixar resto. Submatrizes de diferentes tamanhos podem ser obtidas usando uma matriz de tamanhos de submatrizes. Os elementos da matriz de tamanho são usados até que toda a matriz seja dividida. Se a matriz foi completamente dividida e o tamanho restante não for suficiente para formar uma submatriz completa, o restante não dividido será a última submatriz.

Exemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(5,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(parts,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


/*
false  4003
true  0
splitted 0
[[1,2,3,4,5,6]]
splitted 1
[[7,8,9,10,11,12]]
splitted 2
[[13,14,15,16,17,18]]
splitted 3
[[19,20,21,22,23,24]]
splitted 4
[[25,26,27,28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
[7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
[19,20,21,22,23,24]
[25,26,27,28,29,30]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Vsplit

Divisão vertical de uma matriz em várias submatrizes. O mesmo que Split com axis=1

```
bool matrix::Vsplit(
  const ulong  parts,      // número de submatrizes
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );

void matrix::Vsplit(
  const ulong& parts[],    // tamanhos das submatrizes
  matrix&      splitted[]  // matrizes das submatrizes obtidas
   );
```

Parâmetros

parts

[in] Número de submatrices para divisão da matriz.

splitted

[out] Matriz de submatrices recebidas.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

Se o número de submatrizes for especificado, serão obtidas submatrizes do mesmo tamanho. Em outras palavras, o número de colunas deve ser dividido pelo número de parts sem um resto. Submatrizes de diferentes tamanhos podem ser obtidas usando uma matriz de tamanhos de submatrizes. Os elementos da matriz de tamanho são usados até que toda a matriz seja dividida. Se a matriz foi completamente dividida e o tamanho restante não for suficiente para formar uma submatriz completa, o restante não dividido será a última submatriz.

Exemplo:

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
 matrix_a.Vsplit(parts,splitted);
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);


/*
   splitted 0
   [[1,2,3]
    [7,8,9]
    [13,14,15]]
   splitted 1
   [[4,5,6]
    [10,11,12]
    [16,17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4]
    [9,10]
    [15,16]]
   splitted 2
   [[5,6]
    [11,12]
    [17,18]]

   splitted 0
   [[1,2]
    [7,8]
    [13,14]]
   splitted 1
   [[3,4,5]
    [9,10,11]
    [15,16,17]]
   splitted 2
   [[6]
    [12]
    [18]]

*/
```

© 2000-2024, MetaQuotes Ltd.

ArgSort

Classificação indireta de uma matriz ou vetor.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );

matrix matrix::Sort(
  const int  axis,               // eixo para classificação
  func_name  compare_func=NULL   // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );
```

Parâmetros

axis

[in] Eixo para a realização da classificação: 0 — horizontal, 1 — vertical.

func_name

[in] Comparador. Você pode especificar um dos valores da enumeração ENUM_SORT_MODE ou sua própria função de comparação. Se nenhuma função for especificada, é utilizada a classificação ascendente.

A função de comparação personalizada pode ser de dois tipos:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Neste caso T — tipo de matriz ou vetor, já TContex — tipo de variável context, passado como um parâmetro adicional ao método Sort.

context

[in] Parâmetro opcional adicional que pode ser passado para uma função de classificação personalizada.

Valor retornado

Vetor ou matriz com índices de elementos classificados. Por exemplo, o resultado [4, 2, 0, 1, 3] indica que o elemento na posição zero do vetor ordenado deve ter o índice 4, o elemento na primeira posição deve ter o índice 2 e assim por diante.

© 2000-2024, MetaQuotes Ltd.

Sort

Classificação de uma matriz ou vetor no lugar.

```
void vector::Sort(
  func_name  compare_func=NULL,  // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );

void matrix::Sort(
  func_name  compare_func=NULL   // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );

void matrix::Sort(
  const int  axis,               // eixo para classificação
  func_name  compare_func=NULL   // função de comparação
  T          context             // parâmetro para a função de classificação personal
   );
```

Parâmetros

axis

[in] Eixo para a realização da classificação: 0 — horizontal, 1 — vertical.

func_name

[in] Comparador. Você pode especificar um dos valores da enumeração ENUM_SORT_MODE ou sua própria função de comparação. Se nenhuma função for especificada, é utilizada a classificação ascendente.

A função de comparação personalizada pode ser de dois tipos:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Neste caso T — tipo de matriz ou vetor, já TContex — tipo de variável context, passado como um parâmetro adicional ao método Sort.

context

[in] Parâmetro opcional adicional que pode ser passado para uma função de classificação personalizada.

Valor retornado

Não. A classificação é realizada no local, isto é, aplicada aos dados da matriz/vetor para o qual o método Sort é chamado.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função de classificação                                          |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
  {
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//| Função para executar o script                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- preenchimento do vetor
   vector v(100);
   //--- classificação ascendente
   v.Sort(MyDoubleComparator);   // neste caso é utilizado um parâmetro adicional com
   Print(v);
   // classificação decrescente
   v.Sort(MyDoubleComparator,1); // aqui o parâmetro opcional '1' é explicitamente esp
   Print(v);
  }
```

© 2000-2024, MetaQuotes Ltd.

Operações matemáticas sobre matrizes e vetores

As operações matemáticas de adição, subtração, multiplicação e divisão podem ser realizadas em matrizes e vetores, de forma elementar, isto é, operando-se elemento por elemento.

As funções matemáticas foram originalmente projetadas para realizar operações matemáticas em quantidades escalares. A maioria dessas funções agora também pode ser usada com os novos tipos de dados — com matrizes e vetores — MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. Neste caso, a matriz ou vetor é processado elemento por elemento. Exemplo:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

No caso de MathMod e MathPow como segundo parâmetro, pode ser usado um escalar, uma matriz ou um vetor de tamanho apropriado.

Para ilustrar como calcular o desvio padrão usando funções matemáticas em um vetor, vamos utilizar um exemplo.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- utilizamos uma função de inicialização para preencher o vetor
  vector r(10, ArrayRandom); // csérie de números aleatórios de 0 a 1
//--- calculamos o valor médio
  double avr=r.Mean();       // média da matriz
  vector d=r-avr;            // calculamos a matriz de desvios em relação ao valor méd
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // matriz de quadrados de desvios
  double sum=s2.Sum();       // soma dos quadrados de desvios
//--- calculamos o desvio padrão de 2 maneiras
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("r.Std()=", r.Std());
 }
/*
  avr(r)=0.5300302133243813
  r=[0.8346201971495713,0.8031556138798182,0.6696676534318063,0.05386516922513505,0.54
  d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-0.4761650440992462,0.0190
   std(r)=0.2838269732183663
  r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//| Preenche o vetor com valores aleatórios                          |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

© 2000-2024, MetaQuotes Ltd.

Operações matemáticas

As operações matemáticas de adição, subtração, multiplicação e divisão podem ser realizadas em matrizes e vetores, de forma elementar, isto é, operando-se elemento por elemento.

Ambas as matrizes ou ambos os vetores devem ser do mesmo tipo e ter as mesmas dimensões. Cada membro da matriz opera com o membro correspondente da segunda matriz.

Como segunda soma (multiplicador, subtractor, divisor) também pode ser usado um escalar do tipo apropriado (double, float ou complex). Nesse caso, cada membro da matriz ou vetor operará com o escalar especificado.

```
  matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix matrix_b={{1,2,3},{4,5,6}};

  matrix matrix_c1=matrix_a+matrix_b;
  matrix matrix_c2=matrix_b-matrix_a;
  matrix matrix_c3=matrix_a*matrix_b;   // produto Hadamard (Hadamard product) não dev
  matrix matrix_c4=matrix_b/matrix_a;

  matrix_c1=matrix_a+1;
  matrix_c2=matrix_b-double_value;
  matrix_c3=matrix_a*M_PI;
  matrix_c4=matrix_b/0.1;

//--- operações são localmente possíveis
  matrix_a+=matrix_b;
  matrix_a/=2;
```

As mesmas operações estão disponíveis para vetores.

© 2000-2024, MetaQuotes Ltd.

Funções matemáticas

Série de funções matemáticas: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh - могут быть применены к матрицам и векторам. Neste caso, a matriz ou vetor é processado elemento por elemento.

No caso de MathMod y MathPow como segundo parâmetro, pode ser usado um escalar, uma matriz ou um vetor de tamanho apropriado.

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

© 2000-2024, MetaQuotes Ltd.

Produtos de matrizes e vetores

Os cálculos de matrizes e vetores incluem:

- multiplicação matricial
- multiplicação vetorial
- obtenção da matriz de covariância
- cálculo da correlação cruzada de dois vetores
- cálculo da convolução de dois vetores
- cálculo do coeficiente de correlação

Função	Ação
MatMul	O método MatMul para determinar o produto de matrizes e vetores possui diversas sobrecargas
GeMM	Produto matricial geral de duas matrizes (General Matrix Multiply)
Power	Eleva uma matriz quadrada à potência inteira especificada
Dot	Produto escalar de dois vetores
Kron	Retorna o produto Kronecker de duas matrizes, uma matriz e um vetor, um vetor e uma matriz ou dois vetores
Inner	Produto interno de duas matrizes
Outer	Calcula o produto externo de duas matrizes ou dois vetores
CorrCoef	Calcula o coeficiente de correlação de Pearson (coeficiente de correlação linear)
Cov	Calcula a matriz de covariância
Correlate	Calcula os valores da função de correlação cruzada (correlação cruzada) de dois vetores
Convolve	Retorna a convolução linear discreta de dois vetores de sequência

© 2000-2024, MetaQuotes Ltd.

MatMul

Método MatMul para determinar o produto de matrizes e vetores possui diversas sobrecargas.

Multiplicação de uma matriz por uma matriz: `matrix[M][K] * matrix[K][N] = matrix[M][N]`

```
matrix  matrix::MatMul(
  const matrix&  b      // segunda matriz
   );
```

Multiplicação de um vetor por uma matriz: `horizontal vector[K] * matrix[K][N] = horizontal vector[N]`

```
vector  vector::MatMul(
  const matrix&  b      // matriz
   );
```

Multiplicação de uma matriz por um vetor: `matrix[M][K] * vertical vector[K] = vertical vector[M]`

```
vector  matrix::MatMul(
  const vector&  b      // vetor
   );
```

Multiplicação escalar de vetores: `horizontal vector * vertical vector = dot value`

```
scalar  vector::MatMul(
  const vector&  b      // segundo vetor
   );
```

Parâmetros

b

[in] Matriz ou vetor.

Valor retornado

Matriz, vetor ou escalar, dependendo do método utilizado.

Observação

As matrizes multiplicadas devem ser compatíveis, o número de colunas da primeira matriz deve ser igual ao número de linhas da segunda matriz. A multiplicação de matrizes não é comutativa, a multiplicação da primeira matriz pela segunda não é igual à multiplicação da segunda matriz pela primeira.

O produto matricial consiste em todas as combinações possíveis de produtos escalares dos vetores linha da primeira matriz e dos vetores coluna da segunda matriz.

Com a multiplicação escalar, os vetores devem ter o mesmo comprimento.

Ao multiplicar um vetor e uma matriz, o comprimento do vetor deve corresponder exatamente ao número de colunas da matriz.

© 2000-2024, MetaQuotes Ltd.

Algoritmo ingênuo de multiplicação de matrizes em MQL5:

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

Exemplo de multiplicação de matrizes

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

Exemplo de multiplicação de um vetor horizontal por uma matriz

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criamos uma matriz 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Produto do vetor horizontal v pela matriz m[3,5]");
   Print("Vetor à esquerda v3 = ", v3);
   Print("Matriz à direita m35 = \n", m35);
   Print("v3.MatMul(m35) = vetor horizontal v[5] \n", v3.MatMul(m35));

  /* Resultado
    Produto do vetor horizontal v3 pela matriz m[3,5]
    Vetor à esquerda v3 = [1,2,3]
    Matriz à direita m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = vetor horizontal v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|  Preenchimento da matriz com valores incrementais                |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Exemplo de multiplicação de uma matriz por um vetor vertical

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criamos uma matriz 3x5
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Produto da matriz m[3,5] pelo vetor vertical v[5]");
   vector v5 = {1,2,3,4,5};
   Print("À esquerda m35 = \n",m35);
   Print("À direita v5 = ",v5);
   Print("m35.MatMul(v5) = vetor vertical v[3] \n",m35.MatMul(v5));

  /* Resultado
   Produto da matriz m[3,5] pelo vetor vertical v[5]
   À esquerda m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   À direita v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vetor vertical v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|  Preenchimento da matriz com valores incrementais                |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

Exemplo de multiplicação escalar de vetores

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- multiplicação escalar de vetores horizontais e verticais
   vector a= {1, 2, 3};  // vetor horizontal
   vector b= {4, 5, 6};  // vetor vertical
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- mostramos que o método Dot dá o mesmo resultado
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Resultado
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Ver também

Dot, GeMM

© 2000-2024, MetaQuotes Ltd.

GeMM

O método GeMM determina o produto matricial total de duas matrizes (General Matrix Multiply). Em termos gerais, a expressão é escrita como C ← α A B + β C , onde as matrizes A e B podem ser opcionalmente transpostas. Ao realizar a multiplicação convencional de matrizes AB (MatMul), é assumido que o coeficiente *alpha* é igual a um e o *beta* é igual a zero.

A principal diferença entre GeMM e MatMul em termos de eficiência é que MatMul sempre cria um novo objeto matriz/vetor, enquanto GeMM trabalha com um objeto matriz já existente que não é recriado. Dessa forma, se a memória para a matriz correspondente for alocada com antecedência ao usar o método GeMM e as mesmas dimensões de matriz forem usadas posteriormente, não haverá realocação de memória. Isso pode ser uma vantagem muito importante para o método GeMM em computações em massa, como na otimização em um testador de estratégia ou no treinamento de uma rede neural.

GeMM também possui 4 sobrecargas, assim como o método MatMul. Mas, nesse caso, a semântica da quarta sobrecarga foi alterada para permitir a multiplicação de vetores verticais e horizontais.

Em um objeto de matriz/vetor existente, não é necessário alocar memória para dados com antecedência. Na primeira vez que o GeMM for chamado, a memória será alocada e preenchida com zeros.

Multiplicação de uma matriz por uma matriz: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // primeira matriz
  const matrix &B,     // segunda matriz
  double alpha,        // multiplicador alfa para o produto das matrizes AB
  double beta,         // multiplicador beta para a matriz C
  uint   flags         // combinação de valores de enumeração ENUM_GEMM (bits "ou"), qu
   );
```

Multiplicação de um vetor por uma matriz: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // vetor horizontal
  const matrix &B,     // matriz
  double alpha,        // multiplicador alfa para o produto AB
  double beta,         // multiplicador beta para o vetor C
  uint   flags         // valor da enumeração ENUM_GEMM que determina a transposição da
   );
```

Multiplicação de uma matriz por um vetor: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // matriz
  const vector &B,     // vetor vertical
  double alpha,        // multiplicador alfa para o produto AB
  double beta,         // multiplicador beta para o vetor C
```

© 2000-2024, MetaQuotes Ltd.

```
  uint   flags        // valor da enumeração ENUM_GEMM que determina a transposição de
   );
```

Multiplicação de um vetor por um vetor: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N]`. Esta sobrecarga retorna uma matriz, ao contrário do método MatMul, que retorna um escalar.

```
bool  matrix::GeMM(
  const vector &A,     // primeiro vetor
  const vector &B,     // segundo vetor
  double alpha,        // multiplicador alfa para o produto AB
  double beta,         // multiplicador beta para a matriz C
  uint   flags         // valor da enumeração ENUM_GEMM que determina a transposição da
   );
```

Parâmetros

A

[in] Matriz ou vetor.

B

[in] Matriz ou vetor.

alpha

[in] Multiplicador alfa para o produto AB.

beta

[in] Multiplicador Beta para a matriz resultante C.

flags

[in] Valor da enumeração ENUM_GEMM que determina a transposição das matrizes A, B e C.

Valor retornado

true se a operação for bem sucedida, caso contrário false.

ENUM_GEMM

Enumeração de sinalizadores para o método GeMM.

Identificador	Descrição
TRANSP_A	Uso da matriz transposta A
TRANSP_B	Uso da matriz transposta B
TRANSP_C	Uso da matriz transposta C

Observação

© 2000-2024, MetaQuotes Ltd.

Matrizes e vetores do tipo float, double e complex podem ser usados como parâmetros A e B. Assim, as variantes de modelo do método GeMM têm este aspecto:

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)

bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

Em geral, uma função de multiplicação geral da matriz é descrita como:

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

onde a matriz A é do tamanho M x K, a matriz B é K x N, e a matriz C é M x N.

Assim, as matrizes multiplicadas devem ser compatíveis, o número de colunas da primeira matriz deve ser igual ao número de linhas da segunda matriz. A multiplicação de matrizes não é comutativa, a multiplicação da primeira matriz pela segunda não é igual à multiplicação da segunda matriz pela primeira.

Exemplo:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- calculamos GeMM para dois vetores
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//--- criamos matrizes sob a forma de vetores
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
   matrix_a:
```

© 2000-2024, MetaQuotes Ltd.

```
    [[1]
    [2]
    [3]
    [4]
    [5]]
    matrix_b:
    [[4,3,2,1]]
    */
//-- calculamos GeMM para as duas matrizes e obtemos o mesmo resultado
    matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
    Print("matrix_c:\n ", matrix_c);
    /*
    matrix_c:
     [[4,3,2,1]
     [8,6,4,2]
     [12,9,6,3]
     [16,12,8,4]
     [20,15,10,5]]
    */
   }
```

Ver também

MatMul

© 2000-2024, MetaQuotes Ltd.

Power

Eleva uma matriz quadrada à potência inteira especificada.

```
matrix matrix::Power(
  const int  power      // potência
   );
```

Parâmetros

power

[in] O expoente pode ser qualquer número inteiro: positivo, negativo ou zero.

Valor retornado

Matriz.

Observação

A matriz resultante tem o mesmo tamanho da matriz original. Ao elevar uma matriz a potência 0, o resultado é a matriz identidade. Uma potência positiva n de uma matriz pressupõe que a matriz original é multiplicada por si mesma n vezes. Uma potência negativa -n implica que a matriz original é primeiro invertida e, em seguida, a matriz invertida é multiplicada por si mesma n vezes.

Algoritmo simples para elevar uma matriz a uma potência em MQL5:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- a matriz deve ser quadrática
   if(a.Rows()!=a.Cols())
      return(false);
//--- o tamanho da matriz resultante é exatamente o mesmo
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- ao ser elevada a zero, retorna a matriz identidade
   if(power==0)
      result.Identity();
   else
     {
      //--- se elevada a uma potência negativa, é necessário primeiro inverter a matri
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      else
        {
         result=a;
         for(int i=1; i<power; i++)
            result=result.MatMul(a);
        }
     }
//---
   c=result;
   return(true);
  }
```

Exemplo em MQL5:

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

Exemplo em Python:

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
```

© 2000-2024, MetaQuotes Ltd.

```
i = np.array([[0, 1], [-1, 0]])
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))

i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

© 2000-2024, MetaQuotes Ltd.

Dot

Produto escalar de dois vetores.

```
double vector::Dot(
  const vector&  b       // segundo vetor
   );
```

Parâmetros

b

[in] Vetor.

Valor retornado

Escalar.

Observação

O produto Dot para duas matrizes nada mais é do que o produto matricial matrix::MatMul().

Algoritmo simples para o produto escalar de vetores em MQL5:

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

Exemplo em MQL5:

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

© 2000-2024, MetaQuotes Ltd.

Exemplo em Python:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000-2024, MetaQuotes Ltd.

Kron

Retorna o produto Kronecker de duas matrizes, uma matriz e um vetor, um vetor e uma matriz ou dois vetores.

```
matrix matrix::Kron(
  const matrix&  b      // segunda matriz
   );

matrix matrix::Kron(
  const vector&  b      // vetor
   );

matrix vector::Kron(
  const matrix&  b      // matriz
   );

matrix vector::Kron(
  const vector&  b      // segundo vetor
   );
```

Parâmetros

b

[in] Segunda matriz.

Valor retornado

Matriz.

Observação

O produto Kronecker também é chamado de multiplicação de tensor.

Algoritmo simples do produto Kronecker para duas matrizes em MQL5:

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
```

© 2000-2024, MetaQuotes Ltd.

```
            for(ulong q=0; q<Q; q++)
               matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

   return(matrix_c);
  }
```

Exemplo em MQL5:

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
   [[1,0,2,0,3,0]
    [0,1,0,2,0,3]
    [4,0,5,0,6,0]
    [0,4,0,5,0,6]]

   [[1,2,2,4,3,6]
    [4,8,5,10,6,12]]
  */
```

Exemplo em Python:

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

© 2000-2024, MetaQuotes Ltd.

Inner

Produto interno de duas matrizes.

```
matrix matrix::Inner(
  const matrix&  b      // segunda matriz
   );
```

Parâmetros

b

[in] Matriz.

Valor retornado

Matriz.

Observação

O produto interno de dois vetores nada mais é do que o produto escalar dos dois vetores vector::Dot().

Algoritmo simples do produto interno de duas matrizes em MQL5:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- o número de colunas deve ser o mesmo
   if(a.Cols()!=b.Cols())
      return(false);
//--- o tamanho da matriz resultante depende do número de vetores em cada uma das mat
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Exemplo em MQL5:

```
matrix a={{0,1,2},{3,4,5}};
matrix b={{0,1,2},{3,4,5},{6,7,8}};
matrix c=a.Inner(b);
Print(c);
matrix a1={{0,1,2}};
matrix c1=a1.Inner(b);
Print(c1);

/*
[[5,14,23]
[14,50,86]]
[[5,14,23]]
*/
```

Exemplo em Python:

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

© 2000-2024, MetaQuotes Ltd.

Outer

Calcula o produto externo de duas matrizes ou dois vetores.

```
matrix matrix::Outer(
  const matrix&  b      // segunda matriz
   );

matrix vector::Outer(
  const vector&  b      // segundo vetor
   );
```

Parâmetros

b

[in] Matriz.

Valor retornado

Matriz.

Observação

O produto externo, como o produto Kronecker, também é uma multiplicação em bloco de matrizes (e vetores).

Algoritmo simples para o produto externo de duas matrizes em MQL5:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- tamanho da matriz de resultados depende do tamanho das matrizes
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//---
   return(matrix_c);
  }
```

Exemplo em MQL5:

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[0,0,0,0,0,0,0,0,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

Exemplo em Python:

```
import numpy as np
```

© 2000-2024, MetaQuotes Ltd.

```
A = np.arange(6)
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
```

```
np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

© 2000-2024, MetaQuotes Ltd.

CorrCoef

Calcula o coeficiente de correlação de Pearson (coeficiente de correlação linear).

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // os vetores de observação estão dispostos em filas, cas
);

scalar vector::CorrCoef(
  const vector&  b           // segundo vetor
);
```

Parâmetros

Parâmetros

rowvar

[in] Sinalizador que determina a posição dos vetores na matriz para o cálculo da correlação dos pares - horizontal ou verticalmente. Por padrão *rowvar* é igual a true, significando que os vetores estão nas filas da matriz. Se *rowvar* for igual a false, então as correlações são contadas entre as colunas da matriz.

b

[in] Segundo vetor.

Valor retornado

Valor dos coeficientes de correlação para o vetor.

Matriz quadrada de valores de coeficientes de correlação entre os vetores da matriz para a matriz. O tamanho da matriz corresponde ao número de vetores na matriz, que podem ser dispostos ao longo das linhas (*rowvar*=true) ou por colunas ((*rowvar*=false).

Observação

O coeficiente de correlação está no intervalo [-1, 1].

Devido ao arredondamento de ponto flutuante, a matriz resultante pode não ser hermitiana, os elementos diagonais podem não ser iguais a 1, e os elementos podem não satisfazer a desigualdade abs(a) <= 1. As partes reais e imaginárias são truncadas ao intervalo [-1, 1] para corrigir isto, mas isto nem sempre ajuda em casos complexos.

Algoritmo simples para calcular o coeficiente de correlação de dois vetores em MQL5:

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
      return(0);
```

© 2000-2024, MetaQuotes Ltd.

```
   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
   ymean/=(double)n;

   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//--- cálculo
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- verificação
   if(xv==0 || yv==0)
      return(0);
//--- retornamos o resultado
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

Exemplo em MQL5:

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("vectors correlation ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
   Print("matrix rows correlation\n",matrix_a.CorrCoef());
   matrixf matrix_a2=matrix_a.Transpose();
```

© 2000-2024, MetaQuotes Ltd.

```
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("rows correlation\n",matrix_a3.CorrCoef());
   Print("cols correlation\n",matrix_a3.CorrCoef(false));

  /*
   vectors correlation 0.9149913787841797
   matrix rows correlation
   [[1,0.91499138]
    [0.91499138,1]]
   transposed matrix cols correlation
   [[1,0.91499138]
    [0.91499138,1]]
   rows correlation
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   cols correlation
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

Exemplo em Python:

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("vectors correlation")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("matrix rows correlation")
print(np.corrcoef(ma))
print("transposed matrix cols correlation")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("rows correlation\n",np.corrcoef(ma1))
print("cols correlation\n",np.corrcoef(ma1,rowvar=False))
```

© 2000-2024, MetaQuotes Ltd.

```
transposed matrix cols correlation
[[1.         0.91499142]
 [0.91499142 1.        ]]

rows correlation
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
cols correlation
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

© 2000-2024, MetaQuotes Ltd.

Cov

Calcula a matriz de covariância.

```
matrix matrix::Cov(
  const bool    rowvar=true  // filas ou colunas de vetores de observação
);

matrix vector::Cov(
  const vector&  b      // segundo vetor
);
```

Parâmetros

b

[in] Segundo vetor.

Observação

Calcula a matriz de covariância.

Algoritmo simples para calcular a matriz de covariância de dois vetores em MQL5:

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix& matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
//--- verificação
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_a[i]))
         return(false);
      if(!MathIsValidNumber(vector_b[i]))
         return(false);
     }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//--- cálculo
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
```

© 2000-2024, MetaQuotes Ltd.

```
         matrix_x[i][j]-=t[i];
//--- syrk C=alpha*A^H*A+beta*C (beta=0 and not considered)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
//--- force symmetricity
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

Exemplo em MQL5:

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

Exemplo em Python:

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
```

© 2000-2024, MetaQuotes Ltd.

```
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

© 2000-2024, MetaQuotes Ltd.

Correlate

Calcula os valores da função de correlação cruzada (correlação cruzada) de dois vetores.

```
vector vector::Correlate(
  const vector&           v,         // vetor
  ENUM_VECTOR_CONVOLVE    mode       // modo
   );
```

Parâmetros

v

[in] Segundo vetor.

mode

[in] O parâmetro mode define o modo de cálculo da convolução linear. Valor a partir da enumeração ENUM_VECTOR_CONVOLVE.

Valor retornado

Correlação cruzada de dois vetores.

Observação

O parâmetro mode define o modo de cálculo da convolução linear.

Algoritmo simples para calcular a correlação cruzada de dois vetores em MQL5:

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

Exemplo em MQL5:

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
   full
```

© 2000-2024, MetaQuotes Ltd.

```
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

Exemplo em Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

© 2000-2024, MetaQuotes Ltd.

Convolve

Retorna uma convolução linear discreta de dois vetores de sequência.

```
vector vector::Convolve(
  const vector&            v,         // vetor
  ENUM_VECTOR_CONVOLVE     mode       // modo
   );
```

Parâmetros

v

[out] Segundo vetor.

mode

[in] O parâmetro mode define o modo de cálculo da convolução linear ENUM_VECTOR_CONVOLVE.

Valor retornado

Convolução linear discreta de dois vetores.

Algoritmo simples para calcular a convolução de dois vetores em MQL5:

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
```

© 2000-2024, MetaQuotes Ltd.

```
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

Exemplo em MQL5:

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));
```

© 2000-2024, MetaQuotes Ltd.

```
/*
 full
 [0,1,2.5,4,5.5,7,2.5]
 same
 [1,2.5,4,5.5,7]
 valid
 [2.5,4,5.5]
*/
```

Exemplo em Python:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

© 2000-2024, MetaQuotes Ltd.

Conversão de matrizes

A decomposição de matrizes é uma tarefa que pode surgir nos seguintes casos:

- como uma etapa intermediária no processo de resolução de sistemas de equações lineares
- para consulta de matrizes
- para calcular determinantes
- ao encontrar autovalores e autovetores de uma matriz
- para calcular funções analíticas de matrizes
- ao usar o método dos mínimos quadrados
- na solução numérica de equações diferenciais

Nesse caso, dependendo do problema a ser resolvido, são utilizados vários tipos de decomposição de matrizes.

Função	Ação
Cholesky	Calcula a decomposição de Cholecky.
Eig	Calcula os valores próprios e os vetores próprios direitos de uma matriz quadrada
EigVals	Calcula os valores próprios de uma matriz comum
LU	Executa a decomposição LU de uma matriz como o produto de uma matriz triangular inferior e uma matriz triangular superior
LUP	Executa uma decomposição LUP com rotação parcial, que é uma fatoração LU com apenas permutações de linha: PA=LU
QR	Calcula a fatoração qr de uma matriz
SVD	Calcula a decomposição de valor singular

© 2000-2024, MetaQuotes Ltd.

Cholesky

Calcula a decomposição de Cholesky.

```
bool matrix::Cholesky(
   matrix&  L      // matrix
   );
```

Parâmetros

L

[out] Matriz triangular inferior.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

Retorna a decomposição de Cholesky, L * L.H, de uma matriz quadrada a, onde L é uma matriz triangular inferior e .H é a matriz conjugada transposta (que é a transposição usual quando a é real).a deve ser hermitiana (simétrica, se for real) e positiva definida. A função não verifica se a é hermitiana ou não. Além disso, apenas elementos triangulares e diagonais inferiores a são usados. Na verdade, apenas L é retornado.

Exemplo:

```
matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
matrix matrix_l;
Print("matrix_a\n", matrix_a);

matrix_a.Cholesky(matrix_l);
Print("matrix_l\n", matrix_l);
Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

/*
matrix_a
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
matrix_l
[[2.408279136645086,0]
 [-0.9062640068544704,3.006291985133859]]
check
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
*/
```

© 2000-2024, MetaQuotes Ltd.

Eig

Calcula os autovalores à direita e os autovetores de uma matriz quadrada.

```
bool matrix::Eig(
  matrix&  eigen_vectors,     // matriz autovectorial
  vector&  eigen_values       // vetor de autovalor
   );
```

Parâmetros

eigen_vectors

[out] Matriz de autovetores verticais.

eigen_values

[out] Vetor de autovalores.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
#property script_show_inputs
//--- input parameters
input int  InpSize1   =512;
input int  InpSize2   =256;
input int  InpSize3   =1024;
//+------------------------------------------------------------------+
//| Preenchimento da matriz não degenerada de teste                  |
//+------------------------------------------------------------------+
template<typename T>
void MatrixFill(matrix<T> &matrix_a)
 {
  ulong size_m=matrix_a.Rows();
  ulong size_k=matrix_a.Cols();
  T     value=0.0;
//--- preenchemos a matriz
  for(ulong i=0; i<size_m; i++)
   {
    for(ulong j=0; j<size_k; j++)
     {
      if(i==j)
        matrix_a[i][j]=T(1.0+i);
      else
       {
        value+=1.0;
        matrix_a[i][j]=value;
       }
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int errors=0;

   errors+=TestEigen<double>(InpSize1);
   errors+=TestEigen<double>(InpSize2);
   errors+=TestEigen<double>(InpSize3);

   errors+=TestEigen<float>(InpSize1);
   errors+=TestEigen<float>(InpSize2);
   errors+=TestEigen<float>(InpSize3);
//---
   Print("Test ", errors?"failed":"passed");
   return;
  }

/*
Resultado

Eigen solver of double matrix 512 x 512  passed  vectors=489  time=4268.506 ms
Eigen solver of double matrix 256 x 256  passed  vectors=251  time=417.610 ms
Eigen solver of double matrix 1024 x 1024  passed  vectors=916  time=43708.280 ms
Eigen solver of float matrix 512 x 512  passed  vectors=1  time=2508.357 ms
Eigen solver of float matrix 256 x 256  passed  vectors=1  time=188.859 ms
Eigen solver of float matrix 1024 x 1024  passed  vectors=1  time=27209.666 ms
Test passed
*/

//+------------------------------------------------------------------+
//|  Teste do método Eig                                             |
//+------------------------------------------------------------------+
template<typename T>
int TestEigen(const int size_m)
  {
   int vectors=0;
   matrix<T> matrix_a(size_m, size_m);
   matrix<T> matrix_v(size_m, size_m);
   vector<T> vector_e(size_m);
//--- preenchemos a matriz quadrada
   MatrixFill(matrix_a);
//--- medimos microssegundos
   ulong t1=GetMicrosecondCount();
//--- Eigen Solver
   matrix_a.Eig(matrix_v, vector_e);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- medimos
  ulong t2=GetMicrosecondCount();
//--- verificamos A * v = lambda * v
  for(ulong n=0; n<vector_e.Size(); n++)
   {
    vector<T> eigen_vector=matrix_v.Col(n);
    vector<T> vector_c1   =eigen_vector*vector_e[n];
    vector<T> vector_c2   =matrix_a.MatMul(eigen_vector);

    //--- demasiadas divisões, por isso flexibilizamos a verificação para o 10° dígito
    ulong errors=vector_c1.CompareByDigits(vector_c2, sizeof(T)==sizeof(double) ? 10 :
    if(int(errors)<size_m/10)
      vectors++;
   }
  double elapsed_time=double(t2-t1)/1000.0;
  printf("Eigen solver of %s matrix %d x %d  %s  vectors=%d  time=%.3f ms",
         typename(T), size_m, size_m, vectors>0?"passed":"failed", vectors, elapsed_ti
  return(vectors==0);
 }
```

© 2000-2024, MetaQuotes Ltd.

EigVals

Calcula os valores próprios de uma matriz comum.

```
bool matrix::EigVals(
  vector&  eigen_values      // vetor de autovalores
   );
```

Parâmetros

eigen_values

[out] Vetor de autovalores corretos.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Observação

A única diferença entre EigVals e EigVals é que EigVals não calcula autovetores, apenas autovalores.

© 2000-2024, MetaQuotes Ltd.

LU

Executa a decomposição LU de uma matriz como o produto de uma matriz triangular inferior e uma matriz triangular superior.

```
bool matrix::LU(
  matrix&  L,      // matriz triangular inferior
  matrix&  U       // matriz triangular superior
   );
```

Parâmetros

L

[out] Matriz triangular inferior.

U

[out] Matriz triangular superior.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- decomposição de LU
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- verificamos se está correto A = L * U
   Print("check\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   check
   [[1,2,3,4]
```

© 2000-2024, MetaQuotes Ltd.

```
    [5,2,6,7]
    [8,9,3,10]
    [11,12,14,4]]
  */
```

© 2000-2024, MetaQuotes Ltd.

LUP

Executa uma decomposição LUP com rotação parcial, que é uma fatoração LU com apenas permutações de linha: PA=LU.

```
bool  LUP(
  matrix&  L,      // matriz triangular inferior
  matrix&  U,      // matriz triangular superior
  matrix&  P       // matriz de permutações
   );
```

Parâmetros

L

[out] Matriz triangular inferior.

U

[out] Matriz triangular superior.

P

[out] Matriz de permutações.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- decomposição de LUP
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- verificamos se está correto P * A = L * U
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
```

© 2000-2024, MetaQuotes Ltd.

```
 [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
 [0,0,-7.210526315789473,7.500000000000001]
 [0,0,0,6.697080291970803]]
matrix_p
[[0,0,0,1]
 [0,1,0,0]
 [0,0,1,0]
 [1,0,0,0]]
P * A
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3,10]
 [1,2,3,4]]
L * U
[[11,12,14,4]
 [5,2,6,7]
 [8,9,3.000000000000001,10]
 [1,2,3,4]]
*/
```

© 2000-2024, MetaQuotes Ltd.

QR

Calcula a fatoração qr de uma matriz.

```
bool  QR(
  matrix&  Q,      // matriz com colunas ortonormal
  matrix&  R       // matriz triangular superior
   );
```

Parâmetros

Q

[out] Matriz com colunas ortonormais. Quando o modo complete estiver habilitado, o resultado será uma matriz ortogonal/unitária, dependendo se a é real/complexa. Nesse caso, o determinante pode ser +/- 1. Caso a matriz de entrada possua mais de duas dimensões, será retornado uma pilha de matrizes com as propriedades mencionadas.

R

[out] Matriz triangular superior.

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-0.4082482904638631]
  [0.4625425214347352,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
```

© 2000-2024, MetaQuotes Ltd.

```
[-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
r
[[-1.224744871391589,-0.2415816237971962]
[-1.22474487139159,-1.466326495188786]
[1.224744871391589,1.316496580927726]
[1.224744871391589,0.2415816237971961]]
qr
[[-1.110223024625157e-16,1]
[1,0.9999999999999999]
[1,1]
[2,1]]
*/
```

© 2000-2024, MetaQuotes Ltd.

SVD

Calcula a decomposição de valor singular.

```
bool matrix::SVD(
  matrix&  U,                    // matriz unitária
  matrix&  V,                    // matriz unitária
  vector&  singular_values       // vetor de valores singulares
   );
```

Parâmetros

U

[out] Matriz unitária de ordem m, composta por vetores singulares à esquerda.

V

[out] Matriz unitária de ordem n, composta por vetores singulares à direita.

singular_values

[out] Valores singulares

Valor retornado

Retorna true se bem-sucedido, caso contrário, false.

Exemplo:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//--- fazemos uma decomposição SVD
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// bloco de verificação
//--- U * singular diagonal * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

© 2000-2024, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- mais verificações
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  errors=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.999999999999999]]
```

© 2000-2024, MetaQuotes Ltd.

```
  Ut_U
  [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
   [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
   [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
  vt_V
  [[1,-5.551115123125783e-17,0]
   [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
   [0,1.110223024625157e-16,0.9999999999999996]]
  V_vt
  [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
   [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
   [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
  */
 }
```

© 2000-2024, MetaQuotes Ltd.

Métodos estatísticos

Métodos de obtenção de estatísticas descritivas de matrizes e vetores.

Função	Ação
ArgMax	Retorna o índice do valor máximo
ArgMin	Retorna o índice do valor mínimo
Max	Retorna o valor máximo em uma matriz/vetor
Min	Retorna o valor mínimo em uma matriz/vetor
Ptp	Retorna o intervalo de valores de uma matriz/vetor ou um eixo especificado de uma matriz
Sum	Executa uma soma de elementos de matriz/vetor, que também pode ser realizada ao longo do(s) eixo(s) especificado(s)
Prod	Calcula o produto de elementos de matriz/vetor, que também podem ser executados ao longo do eixo especificado
CumSum	Retorna a soma cumulativa (cumulativa) de elementos de matriz/vetor, incluindo aqueles ao longo do eixo fornecido
CumProd	Retorna o produto cumulativo (cumulativo) dos elementos da matriz/vetor, incluindo aqueles ao longo do eixo fornecido
Percentile	Calcula o percentil especificado dos valores dos elementos de matriz/vetor ou elementos ao longo do eixo especificado
Quantile	Calcula o quantil especificado dos valores dos elementos da matriz/vetor ou dos elementos ao longo do eixo especificado
Median	Calcula a mediana dos elementos de uma matriz/vetor
Mean	Calcula a média aritmética dos valores dos elementos
Average	Calcula a média ponderada dos valores da matriz/vetor
Std	Calcula o desvio quadrático médio (padrão) dos valores dos elementos de uma matriz/vetor ou um eixo especificado
Var	Calcula a variância dos valores dos elementos de uma matriz/vetor
LinearRegression	Calcula o vetor/matriz com os valores de regressão linear calculados

© 2000-2024, MetaQuotes Ltd.

ArgMax

Retorna o índice do valor máximo.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

O índice do valor máximo.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

© 2000-2024, MetaQuotes Ltd.

ArgMin

Retorna o índice do valor mínimo.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Índice de valor mínimo.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Max

Retorna o valor máximo em uma matriz/vetor.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

O valor máximo na matriz/vetor.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Min

Retorna o valor mínimo em uma matriz/vetor.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

O valor mínimo na matriz/vetor.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.Min(0);
   vector rows_min=matrix_a.Min(1);
   double matrix_min=matrix_a.Min();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min value ",matrix_min);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,3,2]
   rows_min=[2,1,4,7]
   min value 1.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Ptp

Retorna a faixa de valores de matriz/vetor ou o eixo de matriz especificado e equivalente ao resultado Max() - Min(). Ptp - Peak to peak, de pico a pico.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Vetor com faixas de valores (máximo - mínimo).

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Sum

Executa uma soma de elementos de matriz/vetor, que também pode ser realizada ao longo do(s) eixo(s) especificado(s).

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Soma de elementos de matriz/vetor, que também pode ser realizada ao longo do(s) eixo(s) especificado(s).

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

© 2000-2024, MetaQuotes Ltd.

Prod

Calcula o produto de elementos de matriz/vetor, que também podem ser executados ao longo do eixo especificado.

```
double vector::Prod(
  const double  initial=1      // multiplicador inicial
   );

double matrix::Prod(
  const double  initial=1      // multiplicador inicial
   );

vector matrix::Prod(
  const int     axis,          // eixo
  const double  initial=1      // multiplicador inicial
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

initial=1

[in] Multiplicador inicial.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.40000000000001]
```

© 2000-2024, MetaQuotes Ltd.

```
   rows_prod=[60,96,120,693]
   prod value 479001600.0
  */
```

© 2000-2024, MetaQuotes Ltd.

CumSum

Retorna a soma cumulativa (cumulativa) de elementos de matriz/vetor, incluindo aqueles ao longo do eixo fornecido.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Soma acumulada de elementos ao longo do eixo dado.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
```

© 2000-2024, MetaQuotes Ltd.

```
 [6,11,15]
 [7,18,27]]
cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
*/
```

© 2000-2024, MetaQuotes Ltd.

CumProd

Retorna o produto cumulativo (cumulativo) dos elementos da matriz/vetor, incluindo aqueles ao longo do eixo fornecido.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal para cada coluna (ou seja, por linhas), 1 — eixo vertical para cada linha (ou seja, por colunas).

Valor retornado

Produto cumulativo de elementos ao longo de um determinado eixo.

Exemplo:

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
```

© 2000-2024, MetaQuotes Ltd.

```
    [1,8,96]
    [6,30,120]
    [7,77,693]]
   cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,4790016
   */
```

© 2000-2024, MetaQuotes Ltd.

Percentile

Calcula o percentil especificado dos valores dos elementos de matriz/vetor ou elementos ao longo do eixo especificado.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // eixo
   );
```

Parâmetros

percent

[in] O percentil a ser calculado deve estar entre 0 e 100, inclusive.

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Percentil: escalar ou vetorial.

Observação

Os valores válidos para o parâmetro percent estão no intervalo [0, 100]. Um algoritmo linear é usado para calcular percentis. Para calcular corretamente os percentis, a sequência deve ser classificada.

Exemplo:

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);
```

© 2000-2024, MetaQuotes Ltd.

```
/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_percentile [5.5,6.5,7.5]
rows_percentile [2,5,8,11]
percentile value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Quantile

Calcula o quantil especificado dos valores dos elementos da matriz/vetor ou dos elementos ao longo do eixo especificado.

```
double vector::Quantile(
  const double quantile      // quantil
   );

double matrix::Quantile(
  const double quantile      // quantil
   );

vector matrix::Quantile(
  const double quantile,     // quantil
  const int    axis          // eixo
   );
```

Parâmetros

quantile

[in] O quantil a ser calculado deve estar entre 0 e 100, inclusive.

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical

Valor retornado

Quantil: escalar ou vetorial.

Observação

A faixa de valores do parâmetro quantil toma valores no intervalo [0,1]. Um algoritmo linear é usado para calcular os quantis. Para calcular corretamente os quantis, a sequência deve ser classificada.

Exemplo:

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
```

© 2000-2024, MetaQuotes Ltd.

```
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_quantile [5.5,6.5,7.5]
rows_quantile [2,5,8,11]
quantile value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Median

Calcula a mediana dos elementos de uma matriz/vetor.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Mediana: escalar ou vetorial.

Observação

A mediana é um valor tal que exatamente metade dos elementos da matriz/vetor será menor que ela, e a outra será maior. O mesmo que Quantile(0,5) e Percentile(50). Para calcular corretamente os percentis, a sequência deve ser classificada.

Exemplo:

```
matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
Print("matrix_a\n",matrix_a);

vectorf cols_median=matrix_a.Median(0);
vectorf rows_median=matrix_a.Median(1);
float matrix_median=matrix_a.Median();

Print("cols_median ",cols_median);
Print("rows_median ",rows_median);
Print("median value  ",matrix_median);


/*
matrix_a
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
cols_median [5.5,6.5,7.5]
```

© 2000-2024, MetaQuotes Ltd.

```
rows_median [2,5,8,11]
median value  6.5
*/
```

© 2000-2024, MetaQuotes Ltd.

Mean

Calcula a média aritmética dos valores dos elementos.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Média aritmética dos valores dos elementos.

Exemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

© 2000-2024, MetaQuotes Ltd.

Average

Calcula a média ponderada dos valores da matriz/vetor.

```
double vector::Average(
  const vector& weigts      // vetor de pesos
   );

double matrix::Average(
  const matrix& weigts      // matriz de pesos
);

vector matrix::Average(
  const matrix& weigts,     // matriz de pesos
  const int     axis        // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Média aritmética: escalar ou vetorial.

Observação

A matriz/vetor de ponderação está associada à matriz/vetor principal.

Exemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

© 2000-2024, MetaQuotes Ltd.

```
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_average [6,6.75,6.75]
rows_average [5,7,5,9]
average value  6.5
*/ value  6.5
```

© 2000-2024, MetaQuotes Ltd.

Std

Calcula o desvio quadrático médio (padrão) dos valores dos elementos de uma matriz/vetor ou um eixo especificado.

```
double vector::Std();

double matrix::Std();

vector matrix::Std(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Desvio padrão: escalar ou vetorial.

Observação

O desvio padrão é a raiz quadrada da média dos quadrados dos desvios em relação à média. Em outras palavras, std = sqrt(mean(x)), onde x = abs(a - a.mean())**2.

O desvio padrão geralmente é calculado como x.sum() / N, onde N = len(x).

Exemplo:

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_std=matrix_a.Std(0);
   vectorf rows_std=matrix_a.Std(1);
   float matrix_std=matrix_a.Std();

   Print("cols_std ",cols_std);
   Print("rows_std ",rows_std);
   Print("std value  ",matrix_std);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
```

© 2000-2024, MetaQuotes Ltd.

```
cols_std [3.2403703,3.0310888,3.9607449]
rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
std value  3.452052593231201
*/
```

© 2000-2024, MetaQuotes Ltd.

Var

Calcula a variância dos valores dos elementos de uma matriz/vetor.

```
double vector::Var();

double matrix::Var();

vector matrix::Var(
  const int  axis      // eixo
   );
```

Parâmetros

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Dispersão: escalar ou vetorial.

Observação

A variância é o desvio padrão da média, ou seja, var = mean(x), onde x = abs(a - a.mean())**2.

A média geralmente é calculada como x.sum() / N, onde N = len(x).

Exemplo:

```
matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
Print("matrix_a\n",matrix_a);

vectorf cols_var=matrix_a.Var(0);
vectorf rows_var=matrix_a.Var(1);
float matrix_var=matrix_a.Var();

Print("cols_var ",cols_var);
Print("rows_var ",rows_var);
Print("var value  ",matrix_var);


/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_var [10.5,9.1875,15.6875]
rows_var [12.666667,20.666666,0.66666669,2.6666667]
```

© 2000-2024, MetaQuotes Ltd.

```
var value  11.916666984558105
*/
```

© 2000-2024, MetaQuotes Ltd.

LinearRegression

Calcula o vetor/matriz com os valores de regressão linear calculados.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // eixo ao longo do qual a regressão é calcula
   );
```

Parâmetros

axis

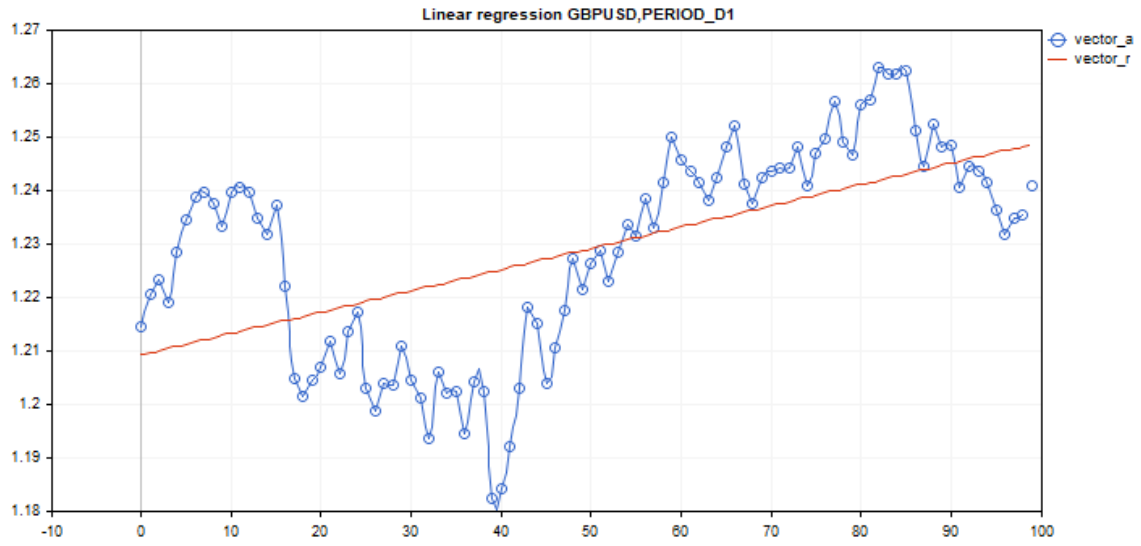
[in] Especifica o eixo ao longo do qual a regressão é contada. Valor a partir da enumeração ENUM_MATRIX_AXIS (AXIS_HORZ — eixo horizontal, AXIS_VERT — eixo vertical).

Valor retornado

Um vetor ou matriz com valores de regressão linear calculados.

Observação

Para calcular uma regressão linear, é usada a equação de regressão padrão: y (x) = a * x + b, onde a é a inclinação da linha e b é seu deslocamento ao longo do eixo Y.



Exemplo:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- switch off chart show
   ChartSetInteger(0,CHART_SHOW,false);

//--- arrays for drawing a graph
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- graph title
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- create graph
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- activation function graph
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//--- derivatives of activation function
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

© 2000-2024, MetaQuotes Ltd.

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- endless loop to recognize pressed keyboard buttons
   while(!IsStopped())
     {
      //--- press escape button to quit program
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- press PdDn to save graph picture
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Save Picture",NULL,"All files (*.*)|*.*",FSD_WRITE_FILE,
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//--- clean up
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Enumeração para indicar o eixo em todas as funções estatísticas para matrizes.

Identificador	Descrição
AXIS_NONE	O eixo não é definido, o cálculo é realizado em todos os elementos da matriz, como se fosse um vetor (consulte o método Flat).
AXIS_HORZ	Eixo horizontal
AXIS_VERT	Eixo vertical

© 2000-2024, MetaQuotes Ltd.

Métodos de Caracterização

Métodos para obter as seguintes características da matriz:

- número de linhas
- numero de colunas
- norma
- número de condicionamento
- determinante
- posto da matriz
- traço
- espectro

Função	Ação
Rows	Retorna o número de linhas em uma matriz
Cols	Retorna o número de colunas em uma matriz
Size	Retorna o tamanho do vetor
Norm	Retorna a norma de uma matriz ou vetor
Cond	Calcula o número de condicionamento de uma matriz
Det	Calcula o determinante de uma matriz quadrada não singular
SLogDet	Calcula o sinal e o logaritmo do determinante de uma matriz
Rank	Retorna a posto de uma matriz usando o método de Gauss
Trace	Retorna a soma dos elementos ao longo das diagonais da matriz
Spectrum	Retorna o espectro de uma matriz como o conjunto de seus autovalores do produto AT*A

© 2000-2024, MetaQuotes Ltd.

Rows

Retorna o número de linhas em uma matriz.

```
ulong matrix::Rows()
```

Valor retornado

Valor do tipo integer.

Exemplo:

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

© 2000-2024, MetaQuotes Ltd.

Cols

Retorna o número de colunas em uma matriz.

```
ulong matrix::Cols()
```

Valor retornado

Valor do tipo integer.

Exemplo:

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

© 2000-2024, MetaQuotes Ltd.

Size

Retorna o tamanho do vetor.

```
ulong vector::Size()
```

Valor retornado

Valor do tipo integer.

Exemplo:

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

© 2000-2024, MetaQuotes Ltd.

Norm

Retorna a norma de uma matriz ou vetor.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,      // norma do vetor
  const int               norm_p=2  // número p-norm caso VECTOR_NORM_P
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm       // norma da matriz
   );
```

Parâmetros

norm

[in] Ordem da norma

Valor retornado

Norma de uma matriz ou vetor.

Observação

- VECTOR_NORM_INF — valor absoluto máximo entre os elementos do vetor.
- VECTOR_NORM_MINUS_INF — valor mínimo absoluto do vetor.
- VECTOR_NORM_P — norma-p do vetor. Se norm_p=0, é o número de elementos não nulos do vetor; norm_p=1 é a soma dos valores absolutos dos elementos vetoriais; norm_p=2 é a raiz quadrada da soma dos quadrados dos valores dos elementos vetoriais. A norma-p pode ser negativa.
- MATRIX_NORM_FROBENIUS — raiz quadrada da soma dos quadrados dos valores dos elementos da matriz. A norma de Frobenius e a norma do vetor P2 são consistentes.
- MATRIX_NORM_SPECTRAL — valor máximo do espectro da matriz.
- MATRIX_NORM_NUCLEAR — soma dos valores singulares da matriz.
- MATRIX_NORM_INF — norma-p1 vetorial máxima entre os vetores verticais da matriz. A norma inf da matriz e a norma inf do vetor são consistentes.
- MATRIX_NORM_MINUS_INF — norma-p1 vetorial mínima entre os vetores verticais da matriz.
- MATRIX_NORM_P1 — norma-p1 vetorial máxima entre os vetores horizontais da matriz.
- MATRIX_NORM_MINUS_P1 — norma-p1 vetorial mínima entre os vetores horizontais da matriz.
- MATRIX_NORM_P2 — maior valor singular da matriz.
- MATRIX_NORM_MINUS_P2 — menor valor singular da matriz.

Algoritmo simples para calcular a norma-p de um vetor em MQL5:

```
double VectorNormP(const vector& v,int norm_value)
  {
   ulong  i;
```

© 2000-2024, MetaQuotes Ltd.

```
   double norm=0.0;
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

Exemplo em MQL5:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
```

© 2000-2024, MetaQuotes Ltd.

```
  [-1,0,1]
  [2,3,4]]
  b.Norm(MATRIX_NORM_P2)=7.745966692414834
  b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
  b.Norm(MATRIX_NORM_INF)9.0
  b.Norm(MATRIX_NORM_MINUS_INF)2.0
  b.Norm(MATRIX_NORM_P1)=)7.0
  b.Norm(MATRIX_NORM_MINUS_P1)=6.0
  b.Norm(MATRIX_NORM_P2)=7.348469228349533
  b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
  */
```

Exemplo em Python:

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))

a
 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

© 2000-2024, MetaQuotes Ltd.

Cond

Calcula o número de condicionamento da matriz.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // norma da matriz
   );
```

Parâmetros

norm

[in] Ordem da norma a partir da enumeração ENUM_MATRIX_NORM.

Valor retornado

Número de condicionamento da matriz. Pode ser infinito.

Observação

O número de condicionamento x é definido como o produto da norma x e seu inverso x [1]. Nesse caso, a norma pode ser a norma L2 usual (raiz da soma dos quadrados) ou uma norma matricial diferente.

O número de condicionamento (ou medida) de uma matriz é o valor K, que é igual ao produto das normas da matriz A e seu inverso. Matrizes com um grande número de condicionamento são chamadas mal condicionadas e vice-versa, matrizes com um pequeno número são chamadas bem condicionadas. A matriz inversa é obtida usando a pseudo-inversão, para não se limitar à condição de que a matriz seja quadrada e não-singular.

Uma exceção é o número de condicionamento espectral.

Algoritmo simples para calcular o número de condicionamento espectral em MQL5:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

  return(norm);
}
```

Exemplo em MQL5:

```
matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

/*
matrix a
[[1,0,-1]
[0,1,0]
[1,0,1]]
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
a.Cond(MATRIX_NORM_INF)=2.0
a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
a.Cond(MATRIX_NORM_P1)=)2.0
a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
*/
```

Exemplo em Python:

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
```

© 2000-2024, MetaQuotes Ltd.

```
print("LA.cond(a, 1)=",LA.cond(a, 1))
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
print("LA.cond(a, -2)=",LA.cond(a, -2))

a
 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

© 2000-2024, MetaQuotes Ltd.

Det

Calcula o determinante de uma matriz quadrada não degenerada

```
double matrix::Det()
```

Valor retornado

Calcula o determinante de uma matriz.

Observação

Determinantes de matrizes de 2ª e 3ª ordens são calculados de acordo com a regra de Sarrus. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

O determinante é calculado pelo método de eliminação de Gauss-Jordan, reduzindo a matriz a uma forma triangular superior. O determinante de uma matriz triangular superior é igual ao produto dos termos da diagonal principal.

Se pelo menos uma linha (coluna) da matriz for zero, o determinante será zero.

Se duas (ou mais) linhas (colunas) de uma matriz são linearmente dependentes, seu determinante será zero.

O determinante da matriz é igual ao produto de seus autovalores.

Exemplo em MQL5:

```
matrix m={{1,2},{3,4}};
double det=m.Det();
Print("matrix m\n",m);
Print("det(m)=",det);
/*
matrix m
[[1,2]
 [3,4]]
det(m)=-2.0
*/
```

Exemplo em Python:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))

a
```

© 2000-2024, MetaQuotes Ltd.

```
 [[1 2]
 [3 4]]

np.linalg.det(a)
 -2.0000000000000004
```

© 2000-2024, MetaQuotes Ltd.

SLogDet

Calcula o sinal e o logaritmo do determinante de uma matriz.

```
double matrix::SLogDet(
  int&  sign      // sinal
   );
```

Parâmetros

sign

[out] O sinal do determinante. Se o valor sign for par, o determinante será positivo.

Valor retornado

Número que representa o sinal do determinante.

Observação

O determinante é calculado pelo método de eliminação de Gauss-Jordan, reduzindo a matriz a uma forma triangular superior. O determinante de uma matriz triangular superior é igual ao produto dos termos da diagonal principal. O logaritmo de um produto é igual à soma dos logaritmos. Por isso, no caso de um estouro ao calcular o determinante, o método SLogDet pode ser usado.

Se o valor sign for par, o determinante será positivo.

Exemplo:

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

© 2000-2024, MetaQuotes Ltd.

Rank

Retorna o posto de uma matriz usando o método de Gauss.

```
int  Rank()
```

Valor retornado

Posto da matriz.

Observação

O posto de um sistema de linhas (colunas) de uma matriz A com m linhas e n colunas é o número máximo de linhas (colunas) linearmente independentes. Várias linhas (colunas) são chamadas linearmente independentes se nenhuma delas puder ser expressa linearmente em termos de outras. O posto do sistema de linhas é sempre igual ao posto do sistema de colunas, e esse número é chamado de posto da matriz.

Exemplo em MQL5:

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;    // déficit da matriz
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimensão - posto 1, a menos que todos sejam "0".

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
```

© 2000-2024, MetaQuotes Ltd.

```
  [0,0,1,0]
  [0,0,0,0]]
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

Exemplo em Python:

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
```

© 2000-2024, MetaQuotes Ltd.

```
matrix_rank(I)= 3

b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000-2024, MetaQuotes Ltd.

Trace

Retorna a soma dos elementos ao longo das diagonais da matriz.

```
double matrix::Trace()
```

Valor retornado

Soma dos elementos ao longo da diagonal.

Observação

O de uma matriz é igual à soma de seus autovalores.

Exemplo em MQL5:

```
matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
a.Reshape(3, 3);
Print("matrix a \n", a);
Print("a.Trace() \n", a.Trace());

/*
matrix a
[[0,1,2]
[3,4,5]
[6,7,8]]
a.Trace()
12.0
*/
```

Exemplo em Python:

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))

a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

© 2000-2024, MetaQuotes Ltd.

Spectrum

Retorna o espectro de uma matriz como um conjunto de seus autovalores do produto AT*A.

```
vector matrix::Spectrum()
```

Valor retornado

Espectro de uma matriz como um vetor de autovalores da matriz.

Exemplo:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

© 2000-2024, MetaQuotes Ltd.

Métodos matriciais para resolver sistemas de equações lineares

Métodos de resolução de sistemas de equações lineares e cálculo da matriz inversa.

Função	Ação
Solve	Resolve uma equação de matriz linear ou um sistema de equações algébricas lineares
LstSq	Resolve um sistema de equações algébricas lineares aproximadamente (para matrizes não quadradas ou degeneradas)
Inv	Calcula uma matriz inversa multiplicativa em relação a uma matriz quadrada não degenerada usando o método Gauss-Jordan
PInv	Calcula o pseudoinverso pelo método Moore-Penrose

© 2000-2024, MetaQuotes Ltd.

Solve

Resolve uma equação de matriz linear ou um sistema de equações algébricas lineares.

```
vector matrix::Solve(
  const vector  b      // valores do ordenado ou "variável dependente".
   );
```

Parâmetros

b

[in] Valores do ordenado ou "variável dependente". (Vetor de membros livres).

Valor retornado

Vetor com solução de sistema a * x = b.

Observação

Se pelo menos uma linha (coluna) da matriz for zero, o sistema não tem solução.

Se duas (ou várias) linhas (colunas) da matriz forem linearmente dependentes, o sistema não tem solução.

Exemplo:

```
//--- solução
   vector_x=matrix_a.Solve(vector_b);
//--- verificamos se está correto a * x = b
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000-2024, MetaQuotes Ltd.

LstSq

Resolve um sistema de equações algébricas lineares aproximadamente (para matrizes não quadradas ou degeneradas).

```
vector matrix::LstSq(
  const vector  b      // valores do ordenado ou "variável dependente".
   );
```

Parâmetros

b

[in] Valores do ordenado ou "variável dependente". (Vetor de membros livres)

Valor retornado

Vetor com solução de sistema a * x = b. Isso é verdade apenas para sistemas que têm uma solução exata.

Exemplo:

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- verificação, deve ser [5, -4]
   Print("x=", x);
//--- verificação, deve ser [7, 40, 3]
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2024, MetaQuotes Ltd.

Inv

Calcula uma matriz inversa multiplicativa em relação a uma matriz quadrada não degenerada usando o método Gauss-Jordan.

```
matrix matrix::Inv()
```

Valor retornado

Matriz inversa multiplicativa.

Observação

O produto da matriz inicial e da matriz inversa é a matriz identidade.

Se pelo menos uma linha (coluna) da matriz for zero, a matriz inversa não pode ser obtida.

Se duas (ou mais) linhas (colunas) de uma matriz são linearmente dependentes, então a matriz inversa não pode ser obtida.

Exemplo:

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- preenchemos a matriz quadrada
   MatrixTestFirst(matrix_a);
//--- medimos microssegundos
   ulong t1=GetMicrosecondCount();
//--- obtemos uma matriz inversa
   matrix inverse=matrix_a.Inv();
//--- medimos
   ulong t2=GetMicrosecondCount();
//--- verificamos se está correto
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- diagonal principal deve conter o valor um
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
```

© 2000-2024, MetaQuotes Ltd.

```
          else
            {
             if(identity[i][j]!=value)
               {
                double diff=MathAbs(identity[i][j]-value);
                //--- muitas multiplicações e divisões, por isso reduzimos a precisão d
                if(diff>1e-9)
                   errors++;
               }
            }
         }
      }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion of matrix %d x %d  %s  errors=%d  time=%.3f ms",size_m,size_m,erı
   return(errors);
  }
```

© 2000-2024, MetaQuotes Ltd.

PInv

Calcula o pseudoinverso pelo método Moore-Penrose.

```
matrix matrix::PInv()
```

Valor retornado

Matriz pseudo-inversa.

Exemplo:

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//--- preenchemos a matriz
   MatrixTestFirst(matrix_a);
//--- invertemos
   matrix_inverted=matrix_a.PInv();
//--- verificamos se está correto
   int errors=0;
//--- A * A+ * A = A   (A+ - matriz pseudo inversa de A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  errors=%d",errors==0?"passed":"fa
//---
   return(errors);
  }
```

© 2000-2024, MetaQuotes Ltd.

Aprendizado de máquina

Métodos para aprendizado de máquina.

A função de ativação de uma rede neural determina o valor de saída de um neurônio dependendo do resultado da soma ponderada das entradas. A escolha da função de ativação tem um grande impacto nas capacidades e no desempenho da rede neural. Diferentes partes do modelo (camadas) podem usar diferentes funções de ativação.

MQL5 implementa não apenas todas as funções de ativação conhecidas, mas também suas derivadas. As funções derivadas permitem que os parâmetros do modelo sejam atualizados eficientemente com base no processo de treinamento.

A tarefa por trás do treinamento de uma rede neural é encontrar um algoritmo que minimize o erro na amostra de treinamento, utilizando a função de perda (loss function). O valor da função perda caracteriza a magnitude do desvio do valor previsto pelo modelo em relação ao real. Dependendo do tipo de problema que está sendo resolvido, diferentes funções de perda são usadas. Por exemplo: para um problema de regressão mean squared error (MSE), para a classificação binária binary cross-entropy (BCE).

Função	Ação
Activation	Calcula os valores da função de ativação e registra no vetor/matriz passado
Derivative	Calcula os valores da derivada da função de ativação e escreve no vetor/matriz passado
Loss	Calcula os valores da função de perda e grava no vetor/matriz passado
LossGradient	Calcula o vetor ou a matriz de gradiente da função de perda
RegressionMetric	Calcula a métrica de regressão como o erro de desvio da linha de regressão criada na matriz de dados especificada
ConfusionMatrix	Calcula a matriz de erros. O método é aplicado a um vetor de valores previstos
ConfusionMatrixMultilabel	Calcula a matriz de erros para cada rótulo. O método é aplicado a um vetor de valores previstos
ClassificationMetric	Calcula uma métrica de classificação para avaliar a qualidade dos dados previstos em relação aos dados reais. O método é aplicado a um vetor de valores previstos
ClassificationScore	Diferentemente de outros métodos na seção Aprendizado de máquina, esse método é aplicado a um vetor de valores reais em vez de um vetor de valores previstos
PrecisionRecall	Calcula os valores para traçar a curva precision-recall. Esse método, assim como o método ClassificationScore, é aplicado a um vetor de valores verdadeiros

© 2000-2024, MetaQuotes Ltd.

Função	Ação
ReceiverOperatingCharacteristic	Calcula os valores para traçar a curva Receiver Operating Characteristic (ROC). Esse método, assim como o método ClassificationScore, é aplicado a um vetor de valores verdadeiros

Exemplo:

Este exemplo mostra o treinamento de um modelo usando operações de matriz. O modelo aprende com a função (a + b + c)^2 / (a^2 + b^2 + c^2). A entrada é uma matriz de dados iniciais, na qual a, b e c estão contidos em colunas separadas. A saída do modelo retorna o resultado da função.

```
matrix weights1, weights2, weights3;                // matrizes de pesos
matrix output1, output2, result;                    // matrizes de saída das camadas ne
input int layer1 = 200;                             // tamanho da 1ª camada oculta
input int layer2 = 200;                             // tamanho da 2ª camada oculta
input int Epochs = 20000;                           // número de épocas de aprendizado
input double lr = 3e-6;                             // taxa de aprendizado
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // função de ativação
//+------------------------------------------------------------------+
//| Função para executar o script                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // tamanho da amostra de treinamento
   int test = 10;       // tamanho da amostra de teste
   matrix m_data, m_target;
//--- geramos uma amostra de treinamento
   if(!CreateData(m_data, m_target, train))
      return;
//--- treinamos o modelo
   if(!Train(m_data, m_target, Epochs))
      return;
//--- geramos uma amostra de teste
   if(!CreateData(m_data, m_target, test))
      return;
//--- teste do modelo
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Método de geração de amostras                                    |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- inicializamos as matrizes de entrada e saída
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- preenchemos a matriz de dados inicial com valores aleatórios
   data.Random(-10, 10);
//--- calculamos os valores-alvo para a amostra de treinamento
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) + MathPow(data.Col(2)
   if(!target.Col(X1 / X2, 0))
      return false;
//--- retornamos o resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de treinamento modelo                                     |
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- criamos o modelo
   if(!CreateNet())
      return false;
//--- treinamos o modelo
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- propagação
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- retropropagação e atualização das matrizes de pesos
      if(!Backprop(data, target))
         return false;
     }
//--- retornamos o resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de criação do modelo                                      |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- inicializamos as matrizes de peso
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- preenchemos as matrizes de peso com valores aleatórios
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- retornamos o resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de propagação                                             |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- verificamos o tamanho dos dados de entrada
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- calculamos a primeira camada neural
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   output1 = temp.MatMul(weights1);
//--- calculamos a função de ativação
   if(!output1.Activation(temp, ac_func))
      return false;
//--- calculamos a segunda camada neural
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- calculamos a função de ativação
   if(!output2.Activation(temp, ac_func))
      return false;
//--- calculamos a terceira camada neural
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- retornamos o resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de retropropagação                                        |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- verificamos a dimensionalidade da matriz de valores-alvo
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- determinamos o desvio dos valores calculados em relação aos valores alvo
   matrix loss = (target - result) * 2;
//--- traçamos um gradiente para a camada anterior
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- atualizamos a matriz de peso da última camada
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
```

© 2000-2024, MetaQuotes Ltd.

```
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- ajustamos o gradiente de erro pela derivada da função de ativação
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- diminuímos o gradiente para uma camada abaixo
   gradient = loss.MatMul(weights2.Transpose());
//--- atualizamos a matriz de peso da 2ª camada oculta
   if(!output1.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- ajustamos o gradiente de erro pela derivada da função de ativação
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- atualizamos a matriz de peso da 1ª camada oculta
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//--- retornamos o resultado
   return true;
  }
//+------------------------------------------------------------------+
//| Método de teste do modelo                                        |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- propagação nos dados de teste
   if(!FeedForward(data))
      return false;
//--- registramos no log os resultados do cálculo do modelo e os valores verdadeiros
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f, Ta
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//--- retornamos o resultado
   return true;
```

© 2000-2024, MetaQuotes Ltd.

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

© 2000-2024, MetaQuotes Ltd.

Activation

Calcula os valores da função de ativação e os grava no vetor/matriz transmitido.

```
bool vector::Activation(
  vector&                    vect_out,      // vetor para obter valores
  ENUM_ACTIVATION_FUNCTION   activation,    // função de ativação
   ...                                      // parâmetros adicionais
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matriz para obter valores
  ENUM_ACTIVATION_FUNCTION   activation     // função de ativação
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // matriz para obter valores
  ENUM_ACTIVATION_FUNCTION   activation,    // função de ativação
  ENUM_MATRIX_AXIS           axis,          // eixo
   ...                                      // parâmetros adicionais
   );
```

Parâmetros

vect_out/matrix_out

[out] Vetor ou matriz para obter os valores calculados da função de ativação.

activation

[in] Função de ativação a partir da enumeração ENUM_ACTIVATION_FUNCTION.

axis

[in] Valor a partir da enumeração ENUM_MATRIX_AXIS (AXIS_HORZ — eixo horizontal, AXIS_VERT — eixo vertical).

...

[in] Parâmetros adicionais necessários para algumas funções de ativação. Se nenhum parâmetro for especificado, serão usados os valores padrão.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false.

Parâmetros adicionais

Algumas funções de ativação aceitam parâmetros adicionais. Se nenhum parâmetro for especificado, serão usados os valores padrão

© 2000-2024, MetaQuotes Ltd.

```
AF_ELU  (Exponential Linear Unit)
  double alpha=1.0

Função de ativação: if(x>=0) f(x) = x
                    else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Função de ativação: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Função de ativação: if(x>=0) f(x)=x
                    else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Função de ativação: if(alpha==0) f(x) = max(x,0)
                    else if(x>max_value) f(x) = x
                    else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

Função de ativação: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Função de ativação: if(x>theta) f(x) = x
                    else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients
```

© 2000-2024, MetaQuotes Ltd.

```
Função de ativação: if(x[i]>=0) f(x)[i] = x[i]
                    else f(x)[i] = alpha[i] * x[i]
```

Observação

Nas redes neurais artificiais, a função de ativação do neurônio determina o sinal de saída, que é determinado pelo sinal de entrada ou pelo conjunto de sinais de entrada. A escolha da função de ativação tem um grande impacto sobre os recursos e o desempenho da rede neural. Diferentes partes do modelo (camadas) podem usar diferentes funções de ativação.

Exemplos do uso de parâmetros adicionais:

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
```

© 2000-2024, MetaQuotes Ltd.

```
   x.Activation(y,AF_SWISH);
   Print(y);
   x.Activation(y,AF_SWISH,2.0);
   Print(y);

   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/*  Resultados
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-0.033464
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-0.000226

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

© 2000-2024, MetaQuotes Ltd.

Derivative

Calcula os valores da derivada da função de ativação e os grava no vetor/matriz transmitido.

```
bool vector::Derivative(
  vector&                    vect_out,      // vetor para obter valores
  ENUM_ACTIVATION_FUNCTION   activation,    // função de ativação
   ...                                      // parâmetros adicionais
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matriz para obter valores
  ENUM_ACTIVATION_FUNCTION   activation,    // função de ativação
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // matriz para obter valores
  ENUM_ACTIVATION_FUNCTION   activation,    // função de ativação
  ENUM_MATRIX_AXIS           axis,          // eixo
   ...                                      // parâmetros adicionais
   );
```

Parâmetros

vect_out/matrix_out

[out] Vetor ou matriz para obter os valores calculados da derivada da função de ativação.

activation

[in] Função de ativação a partir da enumeração ENUM_ACTIVATION_FUNCTION.

axis

[in] Valor a partir da enumeração ENUM_MATRIX_AXIS (AXIS_HORZ — eixo horizontal, AXIS_VERT — eixo vertical).

...

[in] Parâmetros adicionais, que são os mesmos das funções de ativação. Somente algumas funções de ativação aceitam parâmetros adicionais. Se nenhum parâmetro for especificado, serão usados os valores padrão.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false.

Observação

As derivadas das funções permitem que os parâmetros do modelo sejam atualizados com eficiência com base no erro obtido no processo de aprendizado quando o erro se propaga para trás.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Loss

Calcula o valor da função de perda.

```
double vector::Loss(
  const vector&       vect_true,     // vetor de valores reais
  ENUM_LOSS_FUNCTION  loss,          // função de perda
   ...                               // parâmetro adicional
   );


double matrix::Loss(
  const matrix&       matrix_true,   // matriz de valores reais
  ENUM_LOSS_FUNCTION  loss,          // função de perda
   );


double matrix::Loss(
  const matrix&       matrix_true,   // matriz de valores reais
  ENUM_LOSS_FUNCTION  loss,          // função de perda
  ENUM_MATRIX_AXIS    axis,          // eixo
   ...                               // parâmetro adicional
   );
```

Parâmetros

vect_true/matrix_true

[in] Vetor ou matriz de valores reais.

loss

[in] Função de perda a partir da enumeração ENUM_LOSS_FUNCTION.

axis

[in] Valor a partir da enumeração ENUM_MATRIX_AXIS (AXIS_HORZ — eixo horizontal, AXIS_VERT — eixo vertical).

...

[in] Somente a função de perda Huber (LOSS_HUBER) pode ter o parâmetro delta adicional

Valor retornado

Valor double.

Como o parâmetro delta é usado na função de perda Huber (LOSS_HUBER)

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      loss = 0.5 * delta^2 + delta * (error - delta);
```

Observação

A tarefa de treinamento de rede neural é encontrar coeficientes que minimizem o erro na amostra de treinamento, usando a função de perda.

O valor da função de perda descreve a magnitude do desvio entre o valor previsto pelo modelo e o valor real.

Diferentes funções de perda são usadas, dependendo do tipo de problema a ser resolvido. Por exemplo: para um problema de regressão, o erro quadrático médio (mean squared error , MSE), para uma classificação binária, a entropia cruzada binária (binary cross-entropy, BCE).

Exemplo de uma chamada para a função de perda Hubert:

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* Resultado
   0.155
   0.15125
*/
```

© 2000-2024, MetaQuotes Ltd.

LossGradient

Calcula o vetor ou a matriz de gradiente da função de perda.

```
vector vector::LossGradient(
  const vector&       vect_true,      // vetor de valores reais
  ENUM_LOSS_FUNCTION  loss,           // tipo de função de perda
   ...                                // parâmetro adicional
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,    // matriz de valores reais
  ENUM_LOSS_FUNCTION  loss,           // função de perda
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,    // matriz de valores reais
  ENUM_LOSS_FUNCTION  loss,           // função de perda
  ENUM_MATRIX_AXIS    axis,           // eixo
   ...                                // parâmetro adicional
   );
```

Parâmetros

vect_true/matrix_true

[in] Vetor ou matriz de valores reais.

loss

[in] Função de perda a partir da enumeração ENUM_LOSS_FUNCTION.

axis

[in] Valor a partir da enumeração ENUM_MATRIX_AXIS (AXIS_HORZ — eixo horizontal, AXIS_VERT — eixo vertical).

...

[in] Somente a função de perda Huber (LOSS_HUBER) pode ter o parâmetro delta adicional

Valor retornado

Vetor ou matriz de valores de gradiente da função de perda. O gradiente é a derivada parcial sobre dx (x é o valor previsto) da função de perda em um determinado ponto.

Observação

Os gradientes são usados em redes neurais para ajustar os valores da matriz de peso quando o erro se propaga de volta ao treinamento do modelo.

© 2000-2024, MetaQuotes Ltd.

A tarefa de treinamento de rede neural é encontrar coeficientes que minimizem o erro na amostra de treinamento, usando a função de perda.

Diferentes funções de perda são usadas, dependendo do tipo de problema a ser resolvido. Por exemplo: para um problema de regressão, o erro quadrático médio (mean squared error , MSE), para uma classificação binária, a entropia cruzada binária (binary cross-entropy, BCE).

Exemplo de cálculo dos gradientes de uma função de perda

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* Resultado
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

© 2000-2024, MetaQuotes Ltd.

RegressionMetric

Calcula uma métrica de regressão para avaliar a qualidade dos dados previstos em relação aos dados reais

```
double vector::RegressionMetric(
   const vector&           vector_true,   // vetor de valores reais
   ENUM_REGRESSION_METRIC  metric         // tipo de métrica
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // matriz de valores reais
   ENUM_REGRESSION_METRIC  metric         // tipo de métrica
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // matriz de valores reais
   ENUM_REGRESSION_METRIC  metric,        // tipo de métrica
   int                     axis           // eixo
   );
```

Parâmetros

vector_true/matrix_true

[in] Vetor ou matriz de valores reais.

metric

[in] Tipo de métrica a partir da enumeração ENUM_REGRESSION_METRIC.

axis

[in] Eixo. 0 — eixo horizontal, 1 — eixo vertical.

Valor retornado

Métrica calculada — avalia a qualidade dos dados previstos em relação aos dados reais.

Observação

- REGRESSION_MAE — média das diferenças absolutas entre os valores previstos e os valores reais correspondentesи
- REGRESSION_MSE — média dos quadrados das diferenças entre os valores previstos e os valores reais correspondentes
- REGRESSION_RMSE — raiz quadrada do MSE
- REGRESSION_R2 - 1 — MSE(regressão) / MSE(média)
- REGRESSION_MAPE — erro absoluto médio (MAE) em porcentagem
- REGRESSION_MSPE — erro quadrático médio (MSE) como porcentagem
- REGRESSION_RMSLE — RMSE calculado em escala logarítmica

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
   vector y_true = {3, -0.5, 2, 7};
   vector y_pred = {2.5, 0.0, 2, 8};
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* Resultado
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000-2024, MetaQuotes Ltd.

ConfusionMatrix

Calcula a matriz de erros. O método é aplicado a um vetor de valores previstos.

```
matrix vector::ConfusionMatrix(
  const vector&       vect_true,      // vetor de valores reais
   );


matrix vector::ConfusionMatrix(
  const vector&       vect_true,      // vetor de valores reais
   uint                label           // valor do rótulo
   );
```

Parâmetros

vect_true

[in] Vetor de valores reais.

épocas

[in] Valor do rótulo para o cálculo da matriz de erros.

Valor retornado

Matriz de erros. Se o valor da rótulo não for especificado, será retornada uma matriz de erro multiclasse, em que cada rótulo é comparado com cada outro rótulo individualmente. Se um valor de rótulo for especificado, será retornada uma matriz 2 x 2, em que o rótulo especificado é considerado positivo, e todos os outros rótulos são negativos (ovr, one vs rest).

Observação

A matriz de erro C é tal que Cij é igual ao número de observações que estão no grupo i e que também se prevê que estejam no grupo j. Assim, na classificação binária, o número de verdadeiros negativos (TN) é C00, o de falsos negativos (FN) é C10, o de verdadeiros positivos (TP) é C11 e o de falsos positivos (FP) é C01.

Ou seja, essa matriz pode ser representada graficamente dessa forma:

TN	FP
FN	TP

Os tamanhos do vetor de valores reais e do vetor de valores previstos devem ser os mesmos.

Exemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
```

© 2000-2024, MetaQuotes Ltd.

```
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

© 2000-2024, MetaQuotes Ltd.

ConfusionMatrixMultilabel

Calcula a matriz de erros para cada rótulo. O método é aplicado a um vetor de valores previstos.

```
uint vector::ConfusionMatrixMultiLabel(
  const vector&       vect_true,      // vetor de valores reais
   matrix&             confusions[]   // array de matrizes de erro calculadas
   );
```

Parâmetros

vect_true

[in] Vetor de valores reais.

confusions

[out] Array de matrizes 2 x 2 com matrizes de erro calculadas para cada rótulo.

Valor retornado

Tamanho do array de matrizes de erro calculadas. Em caso de falha, é retornado 0

Observação

O array de resultados pode ser dinâmico ou estático. Se o array for estático, ele deverá ter um tamanho pelo menos igual ao número de classes.

Os tamanhos do vetor de valores reais e do vetor de valores previstos devem ser os mesmos.

Exemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
```

© 2000-2024, MetaQuotes Ltd.

```
  [[28,0]
   [1,1]]
  [[24,1]
   [2,3]]
  [[27,1]
   [0,2]]
  [[27,0]
   [2,1]]
  [[25,2]
   [1,2]]
  [[28,0]
   [2,0]]
  [[23,3]
   [0,4]]
*/
```

© 2000-2024, MetaQuotes Ltd.

ClassificationMetric

Calcula uma métrica de classificação para avaliar a qualidade dos dados previstos em relação aos dados reais. O método é aplicado a um vetor de valores previstos.

```
vector vector::ClassificationMetric(
  const vector&        vect_true,      // vetor de valores reais
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   );


vector vector::ClassificationMetric(
  const vector&        vect_true,      // vetor de valores reais
   ENUM_CLASSIFICATION_METRIC metric         // tipo de métrica
   ENUM_AVERAGE_MODE          mode           // modo de média
   );
```

Parâmetros

vect_true

[in] Vetor de valores reais.

metric

[in] Tipo de métrica a partir da enumeração ENUM_CLASSIFICATION_METRIC. São usados valores diferentes de CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC (que são usados no método ClassificationScore).

mode

[in] Modo de cálculo da média a partir da enumeração ENUM_AVERAGE_MODE. Para métricas são usados CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION e CLASSIFICATION_RECALL.

Valor retornado

Vetor que contém a métrica calculada. No caso do modo de cálculo de média AVERAGE_NONE, o vetor contém valores métricos para cada classe sem cálculo de média. (Por exemplo, para classificação binária, seriam 2 métricas para false e true, respectivamente).

Observação sobre os modos de cálculo da média

AVERAGE_BINARY só faz sentido para classificação binária.

AVERAGE_MICRO — calcular métricas globalmente contando o número total de verdadeiros, falsos negativos e falsos positivos.

AVERAGE_MACRO — calcular métricas para cada rótulo e encontrar seu valor médio não ponderado. Isso não leva em conta o desequilíbrio dos rótulos.

AVERAGE_WEIGHTED — calcular as métricas para cada rótulo e encontrar sua média ponderada sobre o suporte (número de instâncias verdadeiras para cada rótulo). Esse cálculo altera a macro para levar em conta o desequilíbrio de rótulo, o que pode levar a uma medida F que não está entre a precisão e a abrangência.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
Print("accuracy=",accuracy);
vector balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY
Print("balanced=",balanced);
Print("");

vector f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO)
Print("f1_micro=",f1_micro);
vector f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO)
Print("f1_macro=",f1_macro);
vector f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WE
Print("f1_weighted=",f1_weighted);
vector f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
Print("f1_none=",f1_none);
Print("");

vector jaccard_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVE
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVE
Print("jaccard_macro=",jaccard_macro);
vector jaccard_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,A
Print("jaccard_weighted=",jaccard_weighted);
vector jaccard_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERA
Print("jaccard_none=",jaccard_none);
Print("");

vector precision_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_micro=",precision_micro);
vector precision_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_macro=",precision_macro);
vector precision_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISI
Print("precision_weighted=",precision_weighted);
vector precision_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,A
Print("precision_none=",precision_none);
Print("");

vector recall_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_micro=",recall_micro);
vector recall_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_macro=",recall_macro);
vector recall_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVE
Print("recall_weighted=",recall_weighted);
```

© 2000-2024, MetaQuotes Ltd.

```
   vector recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE
   Print("recall_none=",recall_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};

   vector f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_
   Print("f1_bin=",f1_bin);
   vector jaccard_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCAR
   Print("jaccard_bin=",jaccard_bin);
   vector precision_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_PREC
   Print("precision_bin=",precision_bin);
   vector recall_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,
   Print("recall_bin=",recall_bin);


/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.66666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.57142

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.571428

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
  recall_bin=[0.4]
*/
```

© 2000-2024, MetaQuotes Ltd.

ClassificationScore

Calcula uma métrica de classificação para avaliar a qualidade dos dados previstos em relação aos dados reais.

Diferentemente de outros métodos na seção Aprendizado de máquina, esse método é aplicado a um vetor de valores reais em vez de um vetor de valores previstos.

```
vector vector::ClassificationScore(
   const matrix&              pred_scores,    // matriz que contém a distribuição de pr
   ENUM_CLASSIFICATION_METRIC metric          // tipo de métrica
   ENUM_AVERAGE_MODE          mode            // modo de média
   );


vector vector::ClassificationScore(
   const matrix&              pred_scores,    // matriz que contém a distribuição de pr
   ENUM_CLASSIFICATION_METRIC metric          // tipo de métrica
   int                        param           // parâmetro adicional
   );
```

Parâmetros

pred_scores

[in] Matriz que contém um conjunto de vetores horizontais com probabilidades para cada classe. O número de linhas da matriz deve corresponder ao tamanho do vetor de valores reais.

metric

[in] Tipo de métrica a partir da enumeração ENUM_CLASSIFICATION_METRIC. São usados os valores CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC.

mode

[in] Modo de cálculo da média a partir da enumeração ENUM_AVERAGE_MODE. Para a métrica são usados CLASSIFICATION_AVERAGE_PRECISION e CLASSIFICATION_ROC_AUC.

param

[in] Para a métrica CLASSIFICATION_TOP_TOP_K_ACCURACY, é necessário especificar um valor K inteiro em vez do modo de cálculo da média.

Valor retornado

Vetor que contém a métrica calculada. No caso do modo de cálculo de média AVERAGE_NONE, o vetor contém valores métricos para cada classe sem cálculo de média. (Por exemplo, para classificação binária, seriam 2 métricas para false e true, respectivamente).

Observação sobre os modos de cálculo da média

AVERAGE_BINARY só faz sentido para classificação binária.

© 2000-2024, MetaQuotes Ltd.

AVERAGE_MICRO — calcula as métricas globalmente, considerando cada elemento da matriz de indicadores de etiquetas como uma etiqueta. A matriz de indicadores de etiquetas é uma matriz com um conjunto de probabilidades para cada etiqueta.

AVERAGE_MACRO — calcular métricas para cada rótulo e encontrar seu valor médio não ponderado. Isso não leva em conta o desequilíbrio dos rótulos.

AVERAGE_WEIGHTED — calcular as métricas para cada rótulo e encontrar sua média ponderada sobre o suporte (número de instâncias verdadeiras para cada rótulo).

Observação

Para a classificação binária, não apenas uma matriz n x 2, em que a primeira coluna contém as probabilidades do rótulo negativo e a segunda coluna contém as probabilidades do rótulo positivo, mas também uma matriz que consiste em uma única coluna com probabilidades positivas pode ser alimentada como entrada. Isso ocorre porque os modelos de classificação binária podem retornar duas probabilidades ou uma probabilidade para um rótulo positivo.

Exemplo:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

//--- label scores          0         1         2         3         4         5
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022, 0.000
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002, 0.000
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015, 0.000
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638, 0.063
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000, 0.000
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001, 0.000
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167, 0.000
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476, 0.000
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910, 0.068
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000, 0.000
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013, 0.000
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071, 0.998
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012, 0.000
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100, 0.000
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035, 0.000
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910, 0.000
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000, 0.000
                    {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000, 0.000
                    {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873, 0.000
                    {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002, 0.000
                    {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678, 0.000
                    {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076, 0.016
                    {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335, 0.036
                    {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112, 0.006
                    {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768, 0.112
                    {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251, 0.000
```

© 2000-2024, MetaQuotes Ltd.

```
                    {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159, 0.001
                    {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003, 0.000
                    {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896, 0.007
                    {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947, 0.473

   vector top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 is in top 2
                    {0.3, 0.4, 0.2},  // 1 is in top 2
                    {0.2, 0.4, 0.3},  // 2 is in top 2
                    {0.7, 0.2, 0.1}}; // 2 isn't in top 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector ap_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score micro = ",ap_micro);
   vector ap_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score macro = ",ap_macro);
   vector ap_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score weighted = ",ap_weighted);
   vector ap_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION
   Print("average precision score none = ",ap_none);
   Print("");

   vector area_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score micro = ",area_micro);
   vector area_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score macro = ",area_macro);
   vector area_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score weighted = ",area_weighted);
   vector area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE
   Print("roc auc score none = ",area_none);
   Print("");

//--- binary classification
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
                        {0.4, 0.6},
                        {0.1, 0.9},
```

© 2000-2024, MetaQuotes Ltd.

```
                        {1.0, 0.0},
                        {0.6, 0.4},
                        {0.8, 0.2},
                        {0.2, 0.8}};

   vector ap=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECIS
   Print("average precision score binary = ",ap);
   vector ap2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVER
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none = [1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none = [1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

© 2000-2024, MetaQuotes Ltd.

PrecisionRecall

Calcula os valores para traçar a curva precision-recall. Esse método, assim como o método ClassificationScore, é aplicado a um vetor de valores verdadeiros.

```
bool vector::PrecisionRecall(
   const matrix&              pred_scores,   // matriz que contém a distribuição de p
   const ENUM_ENUM_AVERAGE_MODE  mode          // modo de cálculo da média
   matrix&                       precision,    // valores de precision calculados pa
   matrix&                       recall,       // valores de recall calculados para
   matrix&                       thresholds,   // limiares classificados em ordem de
   );
```

Parâmetros

pred_scores

[in] Matriz que contém um conjunto de vetores horizontais com probabilidades para cada classe. O número de linhas da matriz deve corresponder ao tamanho do vetor de valores reais.

mode

[in] Modo de cálculo da média a partir da enumeração ENUM_AVERAGE_MODE. São usados apenas AVERAGE_NONE, AVERAGE_BINARY e AVERAGE_MICRO.

precision

[out] Matriz com valores calculados da curva precision. Se não houver cálculo de média (AVERAGE_NONE), o número de linhas na matriz corresponderá ao número de classes de modelo. O número de colunas corresponde ao tamanho do vetor de valores verdadeiros (ou o número de linhas na matriz de distribuição de probabilidade pred_score). No caso da micro-média, o número de linhas na matriz corresponde ao número total de limites, excluindo as duplas.

recall

[out] Matriz com valores calculados da curva recall.

threshold

[out] Matriz de limiar obtida pela classificação da matriz de probabilidade

Observação

Consulte as notas sobre o método ClassificationScore.

Exemplo:

Exemplo de coleta de estatísticas do modelo mnist.onnx (99% de precisão).

```
//--- data for classification metrics
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- input-output
   matrixf image(28,28);
```

© 2000-2024, MetaQuotes Ltd.

```
   vectorf result(10);

//--- testing
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
      //--- collect data
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

Cálculo da precisão

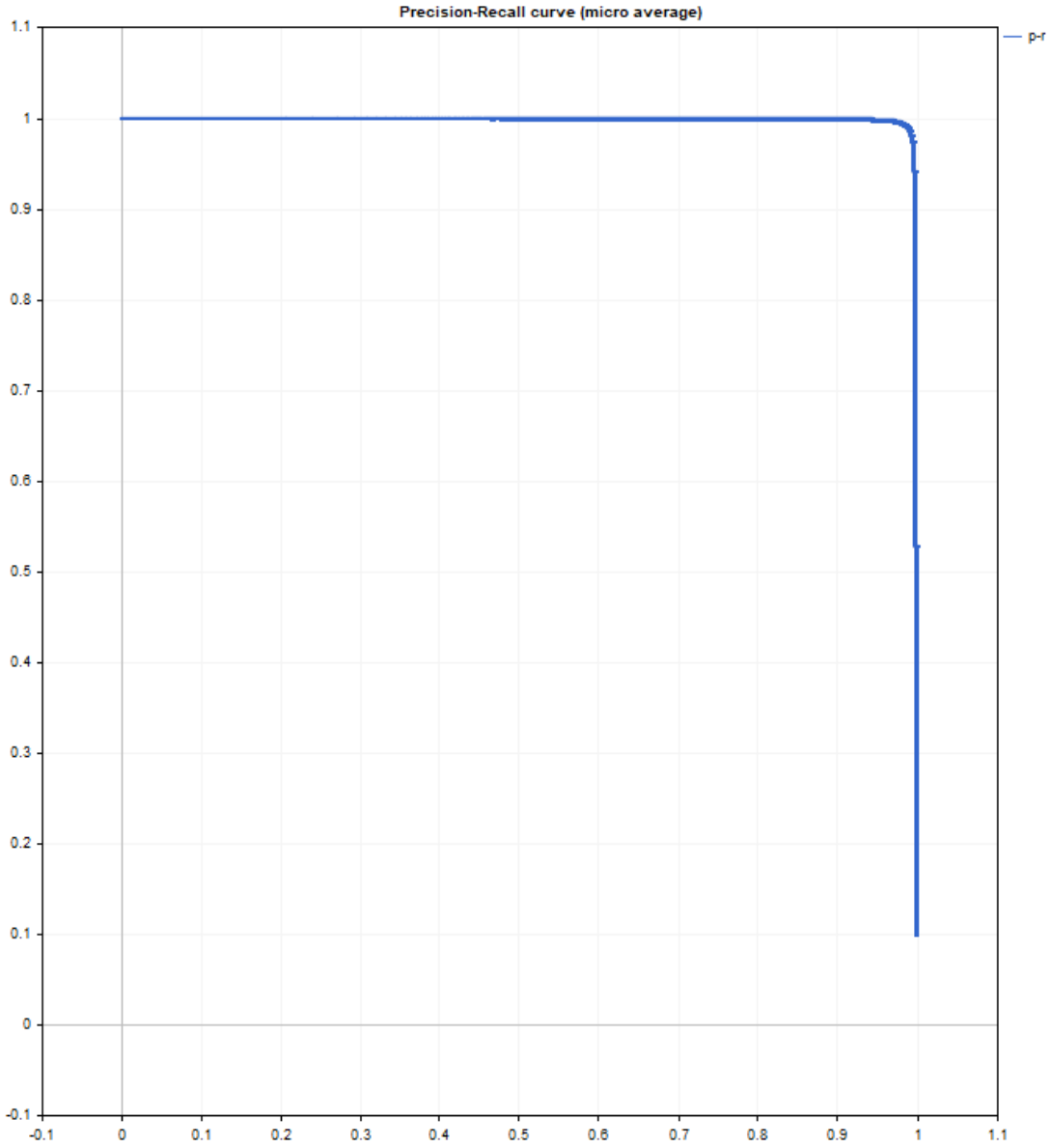
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

Exemplo de geração de gráficos precision-recall, em que os valores de precision são plotados no eixo y e os valores de recall são plotados no eixo x. Assim como a geração de gráficos separados de precision e recall, em que os valores de limite são plotados no eixo x

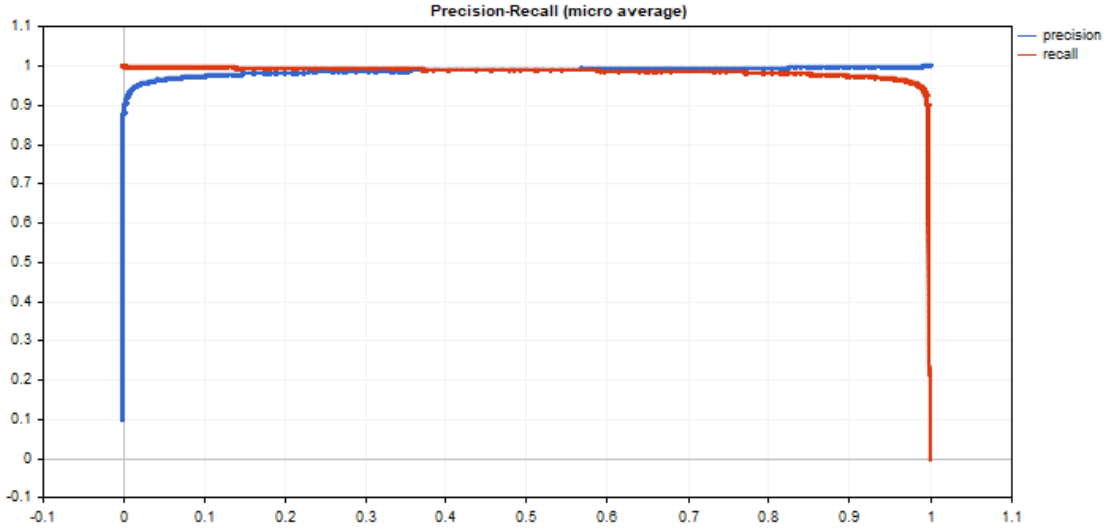
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro average)","precision","recall",thres,precis
     }
```

© 2000-2024, MetaQuotes Ltd.

Resultado:



© 2000-2024, MetaQuotes Ltd.



© 2000-2024, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

Calcula os valores para traçar a curva Receiver Operating Characteristic (ROC). Esse método, assim como o método ClassificationScore, é aplicado a um vetor de valores verdadeiros.

```
bool vector::ReceiverOperatingCharacteristic(
   const matrix&              pred_scores,   // matriz que contém a distribuição de p
   const ENUM_ENUM_AVERAGE_MODE  mode          // modo de cálculo da média
   matrix&                    fpr,           // valores false positive rate calcula
   matrix&                    tpr,           // valores true positive rate calculad
   matrix&                    thresholds,    // limiares classificados em ordem de
   );
```

Parâmetros

pred_scores

[in] Matriz que contém um conjunto de vetores horizontais com probabilidades para cada classe. O número de linhas da matriz deve corresponder ao tamanho do vetor de valores reais.

mode

[in] Modo de cálculo da média a partir da enumeração ENUM_AVERAGE_MODE. São usados apenas AVERAGE_NONE, AVERAGE_BINARY e AVERAGE_MICRO.

fpr

[out] Matriz com valores calculados da curva false positive rate. Se não houver cálculo de média (AVERAGE_NONE), o número de linhas na matriz corresponderá ao número de classes de modelo. O número de colunas corresponde ao tamanho do vetor de valores verdadeiros (ou o número de linhas na matriz de distribuição de probabilidade pred_score). No caso da micro-média, o número de linhas na matriz corresponde ao número total de limites, excluindo as duplas.

tpr

[out] Matriz com valores calculados da curva true positive rate.

threshold

[out] Matriz de limiar obtida pela classificação da matriz de probabilidade

Observação

Consulte as notas sobre o método ClassificationScore.

Exemplo:

Exemplo de exibição de gráfico ROC, em que os valores de tpr são plotados no eixo y e os valores de fpr são plotados no eixo x. Além de gerar gráficos separados de fpr e tpr, em que os valores de limite são plotados no eixo x.

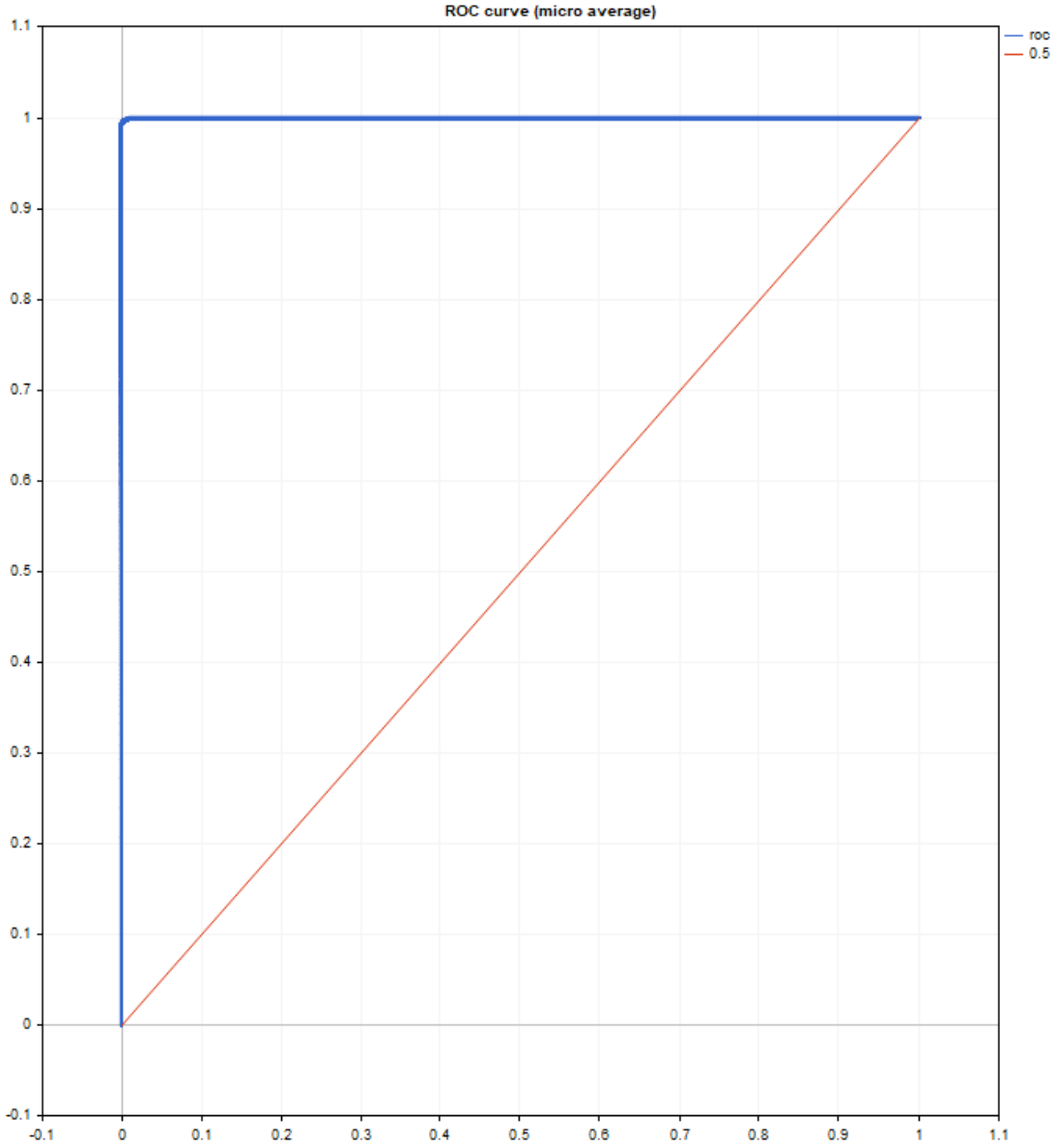
```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;
```

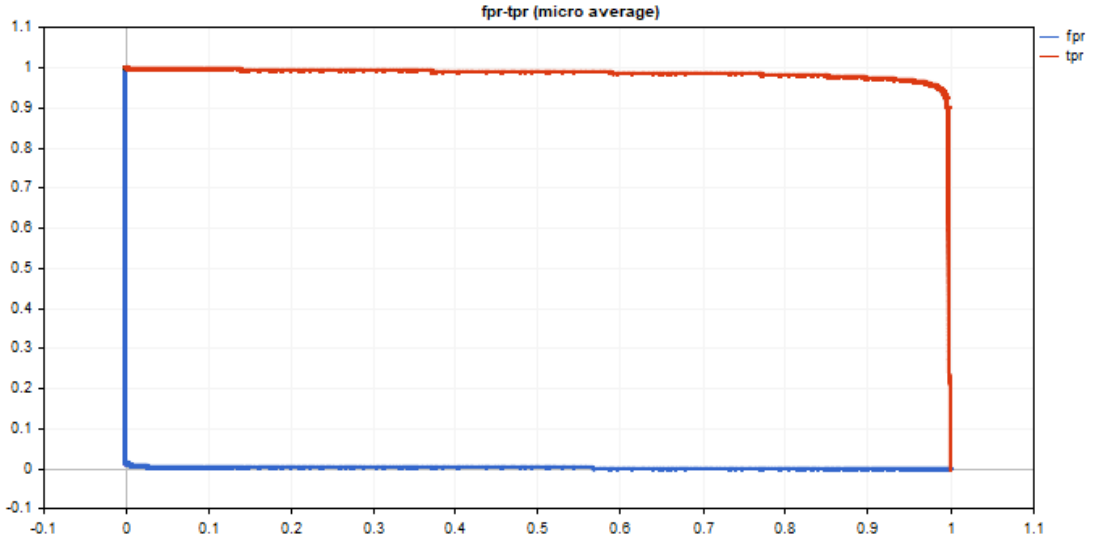
© 2000-2024, MetaQuotes Ltd.

```
    if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,ma
      {
       double fpr[],tpr[],thres[];
       ArrayResize(fpr,mat_thres.Cols());
       ArrayResize(tpr,mat_thres.Cols());
       ArrayResize(thres,mat_thres.Cols());

       for(uint i=0; i<fpr.Size(); i++)
         {
          fpr[i]=mat_fpr[0][i];
          tpr[i]=mat_tpr[0][i];
          thres[i]=mat_thres[0][i];
         }
       thres[0]=thres[1]+0.001;

       PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
       Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
      }
```

Resultado:

© 2000-2024, MetaQuotes Ltd.





© 2000-2024, MetaQuotes Ltd.

O código de geração de gráficos é elementar e se baseia na biblioteca padrão <Graphics/Graphic.mqh>.

São usados dados de teste do modelo mnist.onnx, e o código é apresentado na descrição do método PrecisionRecall.

O ROC AUC está próximo do ideal.

```
roc auc score micro = [0.99991]
```

© 2000-2024, MetaQuotes Ltd.

Funções de Conversão

Este é um grupo de funções que fornece conversão de dados de um formato para outro.

The NormalizeDouble() function must be specially noted as it provides the necessary accuracy of the price presentation. Em operações de negociação, nenhum preço não normalizado deve ser usado caso sua precisão exceder, mesmo que por um dígito, daquilo que é requerida pelo servidor de negociação.

Função	Ação
CharToString	Converte um código de símbolo em um string de um caractere.
DoubleToString	Converte um valor numérico para um linha de texto com a precisão especificada
EnumToString	Converte um valor de enumeração de qualquer tipo para string
NormalizeDouble	Arredonda um número de ponto flutuando para uma precisão especificada
StringToDouble	Converte uma string contendo um representação de símbolo de número em um número de tipo double
StringToInteger	Converte uma string contendo uma representação de símbolo de número em um número de tipo long
StringToTime	Converte uma string contendo hora ou data no formato "yyyy.mm.dd [hh:mi]" em um tipo datetime
TimeToString	Converte um valor contendo hora em segundos decorridos deste 01.01.1970 em uma string de formato "yyyy.mm.dd hh:mi"
IntegerToString	Converte int em uma string de comprimento predefinido
ShortToString	Converte código de símbolo (unicode) em string de um símbolo
ShortArrayToString	Copia parte de um array em uma string
StringToShortArray	Copia símbolo a símbolo uma string para uma parte selecionada de array de tipo ushort
CharArrayToString	Converte código de símbolo (ansi) em um array de um símbolo
StringToCharArray	Copia símbolo a símbolo uma string convertida de Unicode para ANSI, para um lugar selecionado de um array de tipo uchar
CharArrayToStruct	Copia uma matriz do tipo uchar para a estrutura POD
StructToCharArray	Copia a estrutura POD para a matriz do tipo uchar
ColorToARGB	Converte tipo color para tipo uint para receber representação ARGB da cor.
ColorToString	Converte um valor de color em uma string como "R,G,B"
StringToColor	Converte uma string "R,G,B" ou uma string com nome de cor em valor de tipo color

© 2000-2024, MetaQuotes Ltd.

Função	Ação
StringFormat	Converte um número em string de acordo com um formato predefinido

Também Veja

Uso de um Código de Página (Codepage)

© 2000-2024, MetaQuotes Ltd.

CharToString

Converte um código de símbolo em uma string de um caractere.

```
string  CharToString(
   uchar  char_code      // código numérico de um símbolo
   );
```

Parâmetros

char_code

[in] Código de um símbolo ANSI.

Valor do Retorno

String com um símbolo ANSI.

Também Veja

StringToCharArray, ShortToString, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

CharArrayToString

Copia e converte parte de um array de tipo uchar em uma string retornada

```
string  CharArrayToString(
   uchar  array[],              // array
   int    start=0,              // posição de início no array
   int    count=-1              // número de símbolos
   uint    codepage=CP_ACP      // código de página
   );
```

Parâmetros

array[]

[in] Array de tipo uchar.

start=0

[in] Posição a partir da qual a cópia começa. por default 0 é usado.

count=-1

[in] Número de elementos do array para cópia. Define o comprimento de uma string resultante. Valor padrão é -1, que significa copiar até o fim do array, ou até o terminador 0.

codepage=CP_ACP

[in] O valor do código de página. Para os mais usados códigos de página forneça as constantes apropriadas.

Valor do Retorno

String.

Também Veja

StringToCharArray, ShortArrayToString, Uso de um Código de Página (Codepage)

© 2000-2024, MetaQuotes Ltd.

CharArrayToStruct

Copia uma matriz do tipo uchar para a estrutura POD.

```
bool  CharArrayToStruct(
   void&         struct_object,     // estrutura
   const uchar&  char_array[],      // matriz
   uint          start_pos=0        // posição inicial na matriz
   );
```

Parâmetros

struct_object

[in] Referência para qualquer tipo de estrutura POD (estruturas contendo apenas tipos de dados simples).

char_array[]

[in] Matriz do tipo uchar.

start_pos=0

[in] Posição na matriz a partir da qual serão copiados os dados.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Veja também

StringToCharArray, ShortArrayToString, StructToCharArray, Использование кодовой страницы, FileReadStruct, União (union), MathSwap

© 2000-2024, MetaQuotes Ltd.

StructToCharArray

Copia a estrutura POD para a matriz do tipo uchar.

```
bool  StructToCharArray(
   const void&  struct_object,      // estrutura
   uchar&       char_array[],       // matriz
   uint         start_pos=0         // posição inicial na matriz
   );
```

Parâmetros

struct_object

[in] Referência para qualquer tipo de estrutura POD (estruturas contendo apenas tipos de dados simples).

char_array[]

[in] Matriz do tipo uchar.

start_pos=0

[in] Posição na matriz a partir da qual serão copiados os dados.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Observação

Ao copiar, a matriz dinâmica é expandida automaticamente (ArrayResize) se não houver espaço suficiente. Se não for possível expandir a matriz para o valor requerido, a função retornará um erro.

Veja também

StringToCharArray, ShortArrayToString,CharArrayToStruct, Использование кодовой страницы, FileWriteStruct, União (union), MathSwap

© 2000-2024, MetaQuotes Ltd.

ColorToARGB

A função converte o tipo color em tipo uint para obter a representação ARGB da cor. O formato de cor ARGB é usado para gerar recurso gráfico, exibição de texto, assim como a classe da biblioteca padrão CCanvas.

```
uint  ColorToARGB(
   color  clr,          // converte color em formato de cor
   uchar  alpha=255     // canal alfa para gerenciamente da transparência de cor
   );
```

Parâmetros

clr

[in] Valor da cor em variável de tipo color.

alpha

[in] O valor do canal alfa usado para receber a cor em formato ARGB. O valor pode ser definido de 0 (uma cor de um pixel de primeiro plano não altera a exibição de um pixel subjacente) até 255 (uma cor de um pixel subjacente é completamente substituída pela cor do pixel do primeiro plano). Transparência de cor em termos de percentagem é calculada como (1-alpha/255)*100%. Em outras palavras, quanto menor o valor de canal alfa, mais transparente é a cor.

Valor de retorno

Apresenta a cor em formato ARGB onde os valores Alfa, Red, Green, Blue (canal alfa, vermelho, verde, azul) são definidos em séries de quatro bytes de tipo uint.

Observação

RGB é um formato básico e comumente usado para descrever cor de pixel em uma tela em gráficos de computador. Nomes de cores básicas são usadas para atribuir componentes de cor vermelho, verde e azul. Cada componente é descrito por um byte especificando a saturação de cor em uma faixa de 0 a 255 (0x00 a 0XFF formato hexadecimal). Já que a cor branca contém todas as cores, ela é descrita como 0xFFFFFF, ou seja, cada um dos três componentes está presente pelo valor máximo de 0xFF.

Entretanto, algumas tarefas requerem especificar a transparência da cor para descrever a aparência de uma imagem no caso dela ficar coberta, através de uma cor com algum grau de transparência. O conceito de canal alfa é introduzido para tais casos. Ele é implementado como componente adicional do formato RGB. A estrutura do ARGB é mostrado abaixo.

8								8								8								8							
Alpha								Red								Green								Blue							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Valores ARGB são tipicamente expressos usando formato hexadecimal com cada par de dígitos representando os valores dos canais Alpha, Green, e Blue, respectivamente. Por exemplo, a cor 80FFFF00 representa 50.2% amarelo opaco. Inicialmente, 0x80 define valor alfa de 50.2%, uma vez que é 50.2% de valor 0xFF. Então, o primeiro par FF define o valor mais alto do componente vermelho; o próximo par FF é como o anterior mas para o componente verde; e par final 00 representa o valor mais baixo do componente azul que pode haver (ausência de azul). Combinação

© 2000-2024, MetaQuotes Ltd.

de cores verde e vermelha resultam na cor amarela. Se o canal alfa não for usado, a entrada pode ser reduzida para 6 RRGGBB dígitos, é por isso que os valores de canal alfa são armazenados nos bits de topo do tipo inteiro uint.

Dependendo do contexto, dígitos hexadecimais podem ser escritos com o prefixo '0x' ou '#', por exemplo, 80FFFF00, 0x80FFFF00 ou #80FFFF00.

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- define transparência
   uchar alpha=0x55;  // 0x55 significa 55/255=21.6 % de transparência
   //--- fazer conversão de ARGB para cor clrBlue
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB com alpha=0x55 (transparency 21.6%%)",ColorToARG
   //--- fazer conversão de ARGB para cor clrGreen
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB com alpha=0x55 (transparency 21.6%%)",ColorToAR
   //--- fazer conversão de ARGB para cor clrRed
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB com alpha=0x55 (transparency 21.6%%)",ColorToARGB
  }
```

Também Veja

Recursos, ResourceCreate(), TextOut(), Tipo color, Tipos char, short, int e long

© 2000-2024, MetaQuotes Ltd.

ColorToString

Converte valor de cor em string na forma de "R,G,B".

```
string  ColorToString(
   color   color_value,      // valor da cor
   bool    color_name        // mostrar ou não nome da cor
   );
```

Parâmetros

color_value

[in] Valor da cor em variável de tipo color.

color_name

[in] Sinaliza a necessidade de retornar o nome da cor, se o nome da cor for idêntica a uma das constantes de cor predefinidas.

Valor do Retorno

Representação de string da cor como "R,G,B", onde R, G e B são constantes decimais de 0 a 255 convertidos em uma string. Se o parâmetro color_name=true for definido, a função tentará converter o valor da cor no nome da cor.

Exemplo:

```
   string clr=ColorToString(C'0,255,0'); // cor verde
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // obtém a constante de cor
   Print(clr);
```

Também Veja

StringToColor, ColorToARGB

© 2000-2024, MetaQuotes Ltd.

DoubleToString

Converte um valor numérico em string de texto.

```
string  DoubleToString(
   double  value,       // número
   int     digits=8     // numero de dígitos depois do ponto decimal
   );
```

Parâmetros

value

[in] Valor com um ponto flutuante.

digits

[in] Formato de precisão. Se o valor de *digits* estiver na faixa entre 0 e 16, um representação de string de um número com a quantidade especificada de dígitos depois do ponto será obtida. Se o valor de *digits* estiver na faixa entre -1 e -16, uma representação de string de um número em notação científica com a quantidade especificada de dígitos depois do ponto decimal será obtida. Em todos os outros casos, o valor da string conterá 8 dígitos depois do ponto decimal.

Valor do Retorno

String contendo uma representação de um número com a precisão especificada.

Exemplo:

```
   Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

Também Veja

NormalizeDouble, StringToDouble

© 2000-2024, MetaQuotes Ltd.

EnumToString

Converte um valor de enumeração de qualquer tipo para forma de texto.

```
string  EnumToString(
   any_enum  value      // qualquer tipo de valor de enumeração
   );
```

Parâmetros

value

[in] Qualquer tipo de valor de enumeração.

Valor do Retorno

Uma string com uma representação de texto da enumeração. Para obter a mensagem de erro chame a função GetLastError().

Observação

A função pode definir os seguintes valores de erro na variável _LastError:

- ERR_INTERNAL_ERROR - erro do ambiente de execução
- ERR_NOT_ENOUGH_MEMORY - sem memória suficiente para concluir a operação
- ERR_INVALID_PARAMETER - não pode permitir o nome do valor da enumeração

Exemplo:

```
enum interval  // Enumeração de constantes nomeadas
  {
   month=1,     // intervalo de um mês
   two_months,  // dois meses
   quarter,     // três meses - um quadrimestre
   halfyear=6,  // meio ano
   year=12,     // um ano - 12 months
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- define o intervalo de tempo igual a um mês
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- define o intervalo de tempo igual a um quadrimestre (três meses)
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- define o intervalo de tempo igual a um ano (12 meses)
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- verifica como o tipo de ordem é mostrado
```

© 2000-2024, MetaQuotes Ltd.

```
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- verifica como valores incorretos são mostrados
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Resultado:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Também Veja

Enumerações, Variáveis de entrada

© 2000-2024, MetaQuotes Ltd.

IntegerToString

Esta função converte um valor de tipo inteiro em uma string de comprimento especificado e retorna a string obtida.

```
string  IntegerToString(
   long    number,              // número
   int     str_len=0,           // comprimento da string resultado
   ushort  fill_symbol=' '      // símbolo de preenchimento
   );
```

Parâmetros

number

[in] Número para conversão.

str_len=0

[in] Comprimento da string. Se o comprimento da string resultante for maior que aquele especificado, a string não é truncada. Se ela for menor, os símbolos de preenchimento (filler) serão adicionados a esquerda.

fill_symbol=' '

[in] Símbolo de preenchimento. Por padrão, ele é um espaço.

Valor do Retorno

String.

Também Veja

StringToInteger

© 2000-2024, MetaQuotes Ltd.

ShortToString

Converte um código de símbolo (unicode) em uma string de um símbolo e retorna a string resultante.

```
string  ShortToString(
   ushort  symbol_code      // símbolo
   );
```

Parâmetros

symbol_code

[in] Código do símbolo. Ao invés de um código de símbolo, você pode usar uma string literal contendo um símbolo ou uma string literal com código hexadecimal de 2-bytes correspondente ao símbolo da tabela Unicode.

Valor do Retorno

String.

Também Veja

StringToCharArray, CharToString, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

ShortArrayToString

Copia parte de um array em uma string retornada.

```
string  ShortArrayToString(
   ushort  array[],       // array
   int     start=0,       // posição de começo do array
   int     count=-1       // número de símbolos
   );
```

Parâmetros

array[]

[in] Array de tipo ushort (análogo ao tipo wchar_t).

start=0

[in] Posição, a partir da qual a cópia começa, Default - 0.

count=-1

[in] Número de elemento do array para copiar. Define o comprimento de uma string resultante. Valor padrão é -1, que significa copiar até o fim do array, ou até o terminador 0.

Valor do Retorno

String.

Também Veja

StringToShortArray, CharArrayToString, Uso de um Código de Página (Codepage)

© 2000-2024, MetaQuotes Ltd.

TimeToString

Converte um valor contendo hora em segundos decorridos deste 01.01.1970 em uma string de formato "yyyy.mm.dd hh:mi".

```
string  TimeToString(
   datetime  value,                          // número
   int       mode=TIME_DATE|TIME_MINUTES     // formato de saída
   );
```

Parâmetros

value

[in] Hora em segundos a partir de 00:00 1970/01/01.

mode=TIME_DATE|TIME_MINUTES

[in] Modo de entrada de dados adicionais. Pode ser um ou sinalizador combinado:
TIME_DATE obtém como resultado "yyyy.mm.dd",
TIME_MINUTES obtém como resultado "hh:mi",
TIME_SECONDS obêm resultados como "hh:mi:ss".

Valor do Retorno

String.

Também Veja

StringToTime, TimeToStruct

© 2000-2024, MetaQuotes Ltd.

NormalizeDouble

Arredonda um número de ponto flutuante para uma precisão especificada.

```
double  NormalizeDouble(
   double  value,       // número normalizado
   int     digits       // número de dígitos depois do ponto decimal
   );
```

Parâmetros

value

[in] Valor com um ponto flutuante.

digits

[in] Formato de precisão, número de dígitos depois do ponto (0-8).

Valor do Retorno

Valor de tipo double com precisão predefinida.

Observação

Valores calculados de StopLoss, TakeProfit, e valores de preços de abertura para ordens pendentes devem ser normalizados com a precisão obtida através de Digits().

Favor notar que ao imprimir no Diário usando a função Print(), um número normalizado pode conter um número maior de posições decimais do que o esperado. Por exemplo, para:

```
   double a=76.671;              // Um número normalizado com três posições decimais
   Print("Print(76.671)=",a);    // Imprime como é
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // Imprime com um precisão prede
```

você terá o seguinte no terminal:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

Exemplo:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
/*
  Resultado:
```

© 2000-2024, MetaQuotes Ltd.

```
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

Também Veja

DoubleToString, Tipos reais (double, float), Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

StringToCharArray

Copia símbolo a símbolo uma string convertida de Unicode para ANSI, para um lugar selecionado de um array de tipo uchar. Retorna o número de elementos copiados.

```
int  StringToCharArray(
   string  text_string,          // string fonte
   uchar&  array[],              // array
   int     start=0,              // posição de começo no array
   int     count=-1              // número de símbolos
   uint    codepage=CP_ACP       // código de página
   );
```

Parâmetros

text_string

[in] String para copiar.

array[]

[out] Array de tipo uchar.

start=0

[in] Posição a partir da qual a cópia começa. Default - 0.

count=-1

[in] Número de elemento do array para copiar. Define o comprimento de uma string resultante. Valor padrão é -1, que significa copiar até o fim do array, ou até o terminador 0. Terminador 0 será também copiado para o array destino, neste caso o tamanho de um array dinâmico pode ser aumentado se necessário para o tamanho da string. Se o tamanho do array inicializado não é especificado, ele é determinado pelo compilador, baseado no tamanho da seqüência de inicialização.

codepage=CP_ACP

[in] O valor do código de página. Para os mais usados códigos de página forneça as constantes apropriadas.

Valor do Retorno

Número de elementos copiados.

Também Veja

CharArrayToString, StringToShortArray, Uso de um Código de Página (Codepage)

© 2000-2024, MetaQuotes Ltd.

StringToColor

Converte uma string "R,G,B" ou uma string com o nome da cor em valor de tipo color.

```
color  StringToColor(
   string  color_string      // representação de string de cor
   );
```

Parâmetros

color_string

[in] Representação de string de uma cor de tipo "R,G,B" ou nome de uma das Web-cores predefinidas.

Valor do Retorno

Valor da cor.

Exemplo:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- altera a cor um pouco
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Também Veja

ColorToString, ColorToARGB

© 2000-2024, MetaQuotes Ltd.

StringToDouble

A função converte uma string contendo uma representação de um número em um número de tipo double.

```
double  StringToDouble(
   string  value      // string
   );
```

Parâmetros

value

[in] String contém uma representação do símbolo de um número.

Valor do Retorno

Valor de tipo double.

Também Veja

NormalizeDouble, Вещественные типы (double, float), Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

StringToInteger

A função converte uma string contendo uma representação de símbolo de um número em um número de tipo long (inteiro).

```
long  StringToInteger(
   string  value      // string
   );
```

Parâmetros

value

[in] String contendo um número.

Valor do Retorno

Valor de tipo long.

Também Veja

IntegerToString, Вещественные типы (double, float), Conversão de Tipo (Typecasting)

© 2000-2024, MetaQuotes Ltd.

StringToShortArray

A função copia símbolo a símbolo uma string em um lugar especificado de um array de tipo ushort. Retorna o número de elementos copiados.

```
int  StringToShortArray(
   string  text_string,      // string fonte
   ushort& array[],          // array
   int     start=0,          // posição de começo no array
   int     count=-1          // número de símbolos
   );
```

Parâmetros

text_string

[in] String para copiar

array[]

[out] Array de tipo ushort (análogo ao tipo wchar_t).

start=0

[in] Posição, a partir da qual a cópia começa. Default - 0.

count=-1

[in] Número de elemento do array para copiar. Define o comprimento de uma string resultante. Valor padrão é -1, que significa copiar até o final do array, ou até o terminador 0. O terminador 0 será copiado para o array destino, neste caso o tamanho de um array dinâmico pode ser aumentado se necessário para o tamanho da string. Se o tamanho do array inicializado não é especificado, ele é determinado pelo compilador, baseado no tamanho da seqüência de inicialização.

Valor do Retorno

Número de elementos copiados.

Também Veja

ShortArrayToString, StringToCharArray, Uso de um Código de Página (Codepage)

© 2000-2024, MetaQuotes Ltd.

StringToTime

Converte uma string contendo a hora e/ou data no formato "aaaa.mm.dd [hh:mi]" para o tipo datetime.

```
datetime  StringToTime(
   const string  time_string      // string data
   );
```

Parâmetros

time_string

[in] String em um dos formatos especificados:

- "yyyy.mm.dd [hh:mi]"
- "yyyy.mm.dd [hh:mi:ss]"
- "yyyymmdd [hh:mi:ss]"
- "yyyymmdd [hhmiss]"
- "yyyy/mm/dd [hh:mi:ss]"
- "yyyy-mm-dd [hh:mi:ss]"

Valor retornado

O valor do tipo datetime contendo o número de segundos desde 1º de janeiro de 1970.

Observação

Qualquer sequência de tabulação e de espaços entre data e hora é contada como um espaço, de modo que não é necessário nenhum processamento adicional da linha *time_string* antes de chamar a StringToTime().

Veja também

TimeToString, TimeToStruct

© 2000-2024, MetaQuotes Ltd.

StringFormat

A função formata parâmetros obtidos e retorna uma string.

```
string  StringFormat(
   string  format,      // string com descrição de formato
   ...     ...          // parâmetros
   );
```

Parâmetros

format

[in] String contendo método de formatação. As regras de formação são as mesmas da função PrintFormat.

...

[in] Parâmetros, separados por vírgula.

Valor do Retorno

String.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- string variables
   string output_string;
   string temp_string;
   string format_string;
//--- prepare the specification header
   temp_string=StringFormat("Contract specification for %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- int value output
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (number of digits after the decimal
                            digits);
   StringAdd(output_string,temp_string);
//--- double value output with variable number of digits after the decimal point
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (point value)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (current spread in points)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- int value output
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (minimal indention in po
                            min_stop);
   StringAdd(output_string,temp_string);
//--- double value output without the fractional part
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (contract size)\n",
                            contract_size);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (minimal price change)\n",
                            tick_size);
   StringAdd(output_string,temp_string);
//--- determining the swap calculation mode
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (no swa
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (in points)
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST
      case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OP
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CUR
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (by
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2024, MetaQuotes Ltd.

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (long swap value)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- double value output with default accuracy
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (short swap value)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- determining the trading mode
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (tra
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (onl
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (o
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (o
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (no trade re
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (minimal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (minimal volume change step)\n
   StringAdd(output_string,temp_string);
//--- double value output in a compact format
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (maximal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- determining the contract price calculation mode
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";break
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (futures)"
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD for
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (C
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (t
      case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
      case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FU
     }
//--- string value output
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- double value output with 2 digits after the decimal point
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (initial margin)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- double value output with 2 digits after the decimal point
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (maintenance margin)
```

© 2000-2024, MetaQuotes Ltd.

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- int value output
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (order freeze level in
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* execution result
   Contract specification for EURUSD:
     SYMBOL_DIGITS = 5 (number of digits after the decimal point)
     SYMBOL_POINT = 0.00001 (point value)
     SYMBOL_SPREAD = 10 (current spread in points)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (minimal indention in points for Stop orders)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (contract size)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (minimal price change)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (in points)
     SYMBOL_SWAP_LONG = -0.700000 (buy order swap value)
     SYMBOL_SWAP_SHORT = -1.000000 (sell order swap value)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (no trade restrictions)
     SYMBOL_VOLUME_MIN = 0.01 (minimal volume for a deal)
     SYMBOL_VOLUME_STEP = 0.01 (minimal volume change step)
     SYMBOL_VOLUME_MAX = 500 (maximal volume for a deal)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (initial margin)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (maintenance margin)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (order freeze level in points)
*/
  }
```

Também Veja

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2024, MetaQuotes Ltd.

Funções Matemáticas

Um conjunto de funções matemáticas e trigonométricas.

As funções matemáticas foram originalmente projetadas para operações matemáticas em valores escalares. Agora, a maioria dessas funções pode ser usada com novos tipos de dados - matrizes e vetores - MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1 , MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. Neste caso, a matriz ou vetor é processado termo a termo, por exemplo:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

No caso de MathMod e MathPow, um escalar, uma matriz ou um vetor de tamanho apropriado pode ser usado como segundo parâmetro.

Função	Ação
MathAbs	Retorna o valor absoluto (modulo) do valor numérico especificado
MathArccos	Retorna o arco-cosseno de x em radianos
MathArcsin	Retorna o arco-seno de x em radianos
MathArctan	Retorna o arco-tangente de x em radianos
MathArctan2	Retorna em radianos o valor de um ângulo cuja tangente é igual ao quociente de dois números especificados
MathClassify	Retorna o tipo de um número real
MathCeil	Retorna o valor numérico inteiro mais próximo do acima
MathCos	Retorna o cosseno de um número
MathExp	Retorna o exponencial de um número
MathFloor	Retorna o valor numérico inteiro mais próximo do abaixo
MathLog	Retorna o logaritmo natural
MathLog10	Retorna o logaritmo decimal (de um número na base 10)

© 2000-2024, MetaQuotes Ltd.

Função	Ação
MathMax	Retorna o valor máximo entre dois valores numéricos
MathMin	Retorna de valor mínimo entre dois valores numéricos
MathMod	Retorna o resto real da divisão de dois números
MathPow	Eleva uma base a um expoente especificado
MathRand	Retorna um valor pseudo-aleatório entre 0 e 32767
MathRound	Arredonda um valor para o inteiro mais próximo
MathSin	Retorna o seno de um número
MathSqrt	Retorna a raiz quadrada
MathSrand	Define o ponto inicial para geração de uma série de inteiros pseudo-aleatórios
MathTan	Retorna a tangente de um número
MathIsValidNumber	Verifica a validade de um número real
MathExpm1	Retorna o valor da expressão MathExp(x)-1
MathLog1p	Retorna o valor da expressão MathLog(1+x)
MathArccosh	Retorna o valor do arco cosseno hiperbólico
MathArcsinh	Retorna o valor do arco seno hiperbólico
MathArctanh	Retorna o valor do arco tangente hiperbólico
MathCosh	Retorna o valor do cosseno hiperbólico
MathSinh	Retorna o valor do seno hiperbólico
MathTanh	Retorna o valor da tangente hiperbólica
MathSwap	Altera a ordem de bytes em valores do tipo ushort/uint/ushort

© 2000-2024, MetaQuotes Ltd.

MathAbs

A função retorna o valor absoluto (módulo) de um valor numérico especificado.

```
double  MathAbs(
   double  value      // valor numérico
   );
```

Parâmetros

value

[in] Valor numérico.

Valor do Retorno

Valor de tipo double maior ou igual a zero.

Observação

A invés da função MathAbs(), você pode usar fabs().

© 2000-2024, MetaQuotes Ltd.

MathArccos

A função retorna o arco-cosseno de x no intervalo de 0 a π radianos.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Parâmetros

val

[in] Valor de val entre -1 e 1, valor sobre o qual será calculado o arco-cosseno.

Valor do Retorno

Arco-cosseno de um número em radianos. Se val for menor que -1 ou maior que 1, a função retorna NaN (valor indeterminado).

Observação

Ao invés da função MathArccos(), você pode usar acos().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArcsin

A função retorna o arco-seno de x no intervalo de -π/2 a π/2 radianos.

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

Parâmetros

val

[in] Valor de val entre -1 e 1, valor sobre o qual será calculado o arco-seno.

Valor do Retorno

Arco-seno do número val em radianos no intervalo de -π/2 e π/2 radianos. Se val for menor que -1 ou maior que 1, a função retorna NaN (valor indeterminado).

Observação

Ao invés da função MathArcsin(), você pode usar asin().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArctan

A função retorna o arco-tangente de x. Se x for igual a 0, a função retorna 0.

```
double  MathArctan(
   double  value      // tangente
   );
```

Parâmetros

value

[in] Um número representando uma tangente.

Valor do Retorno

MathArctan retorna um valor no intervalo de -π/2 a π/2 radianos.

Observação

Ao invés da função MathArctan(), você pode usar atan().

© 2000-2024, MetaQuotes Ltd.

MathArctan2

Retorna em radianos o valor de um ângulo cuja tangente é igual ao quociente de dois números especificados.

```
double  MathArctan2(
   double  y      // coordenada y do ponto
   double  x      // coordenada x do ponto
   );
```

Parâmetros

y

[in] Número representando a coordenada Y.

x

[in] Número representando a coordenada X.

Valor retornado

MathArctan2 retorna o valor do ângulo θ no intervalo de -π a π radianos, tal que MathTan(θ)=y/x.

Observe o seguinte:

- Para (x, y) no quadrante 1, 0 < θ < π/2
- Para (x, y) no quadrante 2, π/2 < θ≤π
- Para (x, y) no quadrante 3, -π < θ < -π/2
- Para (x, y) no quadrante 4, -π/2 < θ < 0

Para os pontos fora dos quadrantes indicados, o valor retornado é especificado abaixo:

- Se y é igual a 0 e x não é negativo, θ = 0.
- Se y é igual a 0 e x é negativo, θ = π.
- Se y é um número positivo, e x é igual a 0, θ = π/2.
- Se y é um número negativo, e x é igual a 0, θ = -π/2.
- Se y é igual a 0 e x é igual a 0, θ = 0.

Observação

Em vez da função MathArctan2(), pode ser usada a função atan2().

© 2000-2024, MetaQuotes Ltd.

MathClassify

Determina o tipo de um número real e retorna o resultado como um valor da enumeração ENUM_FP_CLASS

```
ENUM_FP_CLASS  MathClassify(
   double  value      // número real
   );
```

Parâmetros

value

[in] Número real verificado

Valor retornado

Valor a partir da enumeração ENUM_FP_CLASS

ENUM_FP_CLASS

Identificador	Descrição
FP_SUBNORMAL	Número subnormal que está mais próximo de zero do que do menor número normalizado representável DBL_MIN (2.2250738585072014e-308)
FP_NORMAL	Número normalizado que varia de 2,2250738585072014e-308 a 1,7976931348623158e+308
FP_ZERO	Zero positivo ou negativo
FP_INFINITE	Número que não pode ser representado pelo tipo correspondente - infinito positivo ou negativo
FP_NAN	Não é um número

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- test NaN
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- test infinity
  double inf=double("inf");
```

© 2000-2024, MetaQuotes Ltd.

```
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
              inf,
              EnumToString(MathClassify(inf)),
              (string)MathIsValidNumber(inf));
//--- test normal value
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
              normal,
              EnumToString(MathClassify(normal)),
              (string)MathIsValidNumber(normal));
//--- test subnormal value
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
              sub_normal,
              EnumToString(MathClassify(sub_normal)),
              (string)MathIsValidNumber(sub_normal));
//--- test zero value
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
              zero,
              EnumToString(MathClassify(zero)),
              (string)MathIsValidNumber(zero));
 }
 /*
 Result:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

Veja também

Tipos reais (double, float), MathIsValidNumber

© 2000-2024, MetaQuotes Ltd.

MathCeil

A função retorna o valor do número inteiro mais próximo do acima.

```
double  MathCeil(
   double  val      // número
   );
```

Parâmetros

val

[in] Valor numérico.

Valor do Retorno

Valor numérico representando o menor inteiro que excede ou é igual a val.

Observação

Ao invés da função MathCeil(), você pode usar ceil().

© 2000-2024, MetaQuotes Ltd.

MathCos

A função retorna o cosseno de um ângulo.

```
double  MathCos(
   double  value      // número
   );
```

Parâmetros

value

[in] Ângulo em radianos.

Valor do Retorno

Valor de tipo double no intervalo de -1 a 1.

Observação

A invés de MathCos(), você pode usar cos().

© 2000-2024, MetaQuotes Ltd.

MathExp

A função retorna o valor e elevado a potência d.

```
double  MathExp(
   double  value      // potência para o número e
   );
```

Parâmetros

value

[in] Um número especificando a potência.

Valor do Retorno

Um número de tipo double. Em "overflow", a função retorna INF (infinito), em caso de perda de ordem MathExp retorna 0.

Observação

Ao invés de MathExp(), você pode usar exp().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathFloor

A função retorna o valor numérico inteiro mais próximo do abaixo.

```
double  MathFloor(
   double  val     // número
   );
```

Parâmetros

val

[in] Valor numérico.

Valor do Retorno

Um valor numérico representando o maior inteiro que é menor ou igual a val.

Observação

Ao invés de MathFloor(), você pode usar floor().

© 2000-2024, MetaQuotes Ltd.

MathLog

A função retorna o logaritmo natural.

```
double  MathLog(
   double  val      // valor aplicado ao logaritmo
   );
```

Parâmetros

val

[in] Valor cujo logaritmo deve ser encontrado.

Valor do Retorno

O logaritmo natural de val em caso de sucesso. Se val for negativo, a função retorna NaN (valor indeterminado). Se val for igual a 0, a função retorna INF (infinito).

Observação

Ao invés de MathLog(), você pode usar log().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathLog

Retorna o logaritmo de um número na base 10.

```
double  MathLog10(
   double  val      // número aplicado ao logaritmo
   );
```

Parâmetros

val

[in] Valor numérico cujo logaritmo comum deve ser encontrado.

Valor do Retorno

O logaritmo comum em caso de sucesso. Se val for negativo, a função retorna NaN (valor indeterminado). Se val for igual a 0, a função retorna INF (infinito).

Observação

Ao invés de MathLog10(), você pode usar log10().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathMax

A função retorna o valor máximo entre dois valores.

```
double  MathMax(
   double  value1,      // primeiro valor
   double  value2       // segundo valor
   );
```

Parâmetros

value1

[in] Primeiro valor numérico.

value2

[in] Segundo valor numérico.

Valor do Retorno

O maior dos dois valores.

Observação

Ao invés de MathMax(), você pode usar fmax(). As funções fmax(), fmin(), MathMax(), MathMin() podem trabalhar com tipos inteiros sem convertê-los para o tipo double.

Se parâmetros de diferentes tipos são passados para um função, o parâmetro do menor tipo é automaticamente convertido para o maior tipo. O tipo do valor de retorno corresponde ao do maior tipo.

Se dados de mesmo tipo são passados, nenhuma conversão é realizada.

© 2000-2024, MetaQuotes Ltd.

MathMin

A função retorna o valor mínimo entre dois valores.

```
double  MathMin(
   double  value1,      // primeiro valor
   double  value2       // segundo valor
   );
```

Parâmetros

value1

[in] Primeiro valor numérico.

value2

[in] Segundo valor numérico.

Valor do Retorno

O menor valor dos dois valores.

Observação

Ao invés de MathMin(), você pode usar fmin(). As funções fmax(), fmin(), MathMax(), MathMin() podem trabalhar com tipos inteiros sem convertê-los para o tipo double.

Se parâmetros de diferentes tipos são passados para um função, o parâmetro do menor tipo é automaticamente convertido para o maior tipo. O tipo do valor de retorno corresponde ao do maior tipo.

Se dados de mesmo tipo são passados, nenhuma conversão é realizada.

© 2000-2024, MetaQuotes Ltd.

MathMod

A função retorna o resto real da divisão de dois números.

```
double  MathMod(
   double  value,       // valor do dividendo
   double  value2       // valor do divisor
   );
```

Parâmetros

value

[in] Valor do dividendo.

value2

[in] Valor do divisor.

Valor do Retorno

A função MathMod calcula o valor real de f de val / y tal que val = i * y + f , onde i é um inteiro, f tem o mesmo sinal de val, e o valor absoluto de f é menor que o valor absoluto de y.

Observação

Ao invés de MathMod(), você pode usar fmod().

© 2000-2024, MetaQuotes Ltd.

MathPow

A função eleva uma base a uma potência especificada.

```
double  MathPow(
   double  base,          // base
   double  exponent       // valor do expoente
   );
```

Parâmetros

base

[in] Base.

exponent

[in] Valor do expoente.

Valor do Retorno

Valor da base elevada a uma potência especificada.

Observação

Ao invés de MathPow(), você pode usar pow().

© 2000-2024, MetaQuotes Ltd.

MathRand

Retorna um inteiro pseudo-aleatório no intervalo de 0 a 32767.

```
int  MathRand();
```

Valor do Retorno

Valor de um inteiro no intervalo de 0 a 32767.

Observação

Antes da primeira chamada da função, é necessário chamar MathSrand para definir o gerador de números pseudo-aleatório para o estado inicial.

Observação

Ao invés de MathRand(), você pode usar rand().

© 2000-2024, MetaQuotes Ltd.

MathRound

A função retorna o valor arredondado para o inteiro mais próximo de um valor numérico especificado.

```
double  MathRound(
   double  value      // valor a ser arredondado
   );
```

Parâmetros

value

[in] Valor numérico antes do arredondamento.

Valor do Retorno

Valor arredondado até o inteiro mais próximo.

Observação

Ao invés de MathRound(), você pode usar round().

© 2000-2024, MetaQuotes Ltd.

MathSin

Retorna o seno de um ângulo especificado.

```
double  MathSin(
   double  value      // argumento em radianos
   );
```

Parâmetros

value

[in] Ângulo em radianos.

Valor do Retorno

Seno de um ângulo medido em radianos. Retorna um valor no intervalo de -1 a 1.

Observação

Ao invés de MathSin(), você pode usar sin().

© 2000-2024, MetaQuotes Ltd.

MathSqrt

Retorna a raiz quadrada de um número.

```
double  MathSqrt(
   double  value      // número positivo
   );
```

Parâmetros

value

[in] Valor numérico positivo.

Valor do Retorno

Raiz quadrada de um valor. Se val for negativo, a função retorna NaN (valor indeterminado).

Observação

Ao invés de MathSqrt(), você pode usar sqrt().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathSrand

Define o ponto inicial para geração de uma série de inteiros pseudo-aleatórios.

```
void  MathSrand(
   int  seed      // número de inicialização
   );
```

Parâmetros

seed

[in] Número inicial para a seqüência de números aleatórios.

Valor do Retorno

Sem valor de retorno.

Observação

A função MathRand() é usada para gerar uma seqüência de números pseudo-aleatórios. Chamar MathSrand() com um certo número de inicialização permite produzir sempre a mesma seqüência de números pseudo-aleatórios.

Para garantir o recebimento de uma seqüência não recorrente, use a chamada de MathSrand(GetTickCount()), uma vez que o valor de GetTickCount() aumenta a partir do momento do início do sistema operativo e não se repete dentro de 49 dias, quando o contador built-in de milissegundos transborda. O uso do MathSrand(TimeCurrent()) não é adequado, porque a função TimeCurrent() retorna a hora do último tick, que pode não se alterar por um longo tempo, por exemplo, nos finais de semana.

A inicialização do gerador de números aleatórios usando MathSrand() para indicadores e Expert Advisors é melhor realizada no handler OnInit(); isso evita os seguidos reinícios múltiplos caso a inicialização do gerador fosse colocada em OnTick() ou OnCalculate().

Ao invés da função MathSrand(), você pode usar a função srand().

Exemplo:

```
#property description "O indicador mostra o teorema do limite central, que:"
#property description "A soma de um número suficientemente grande de variáveis aleató
#property description "A soma de um número suficientemente grande de variáveis aleató
#property description "ou faz uma contribuição determinante para a soma), tem uma dis

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Propriedades da construção gráfica
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- Uma variável de entrada
input int      sample_number=10;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Um buffer do indicador para desenhar a distribuição
double         LabelBuffer[];
//--- Um contador de ticks
double         ticks_counter;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Ligando um array e um buffer de indicador
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- transformar o buffer do indicador em torno do presente para o passado
   ArraySetAsSeries(LabelBuffer,true);
//--- Inicializar o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- Inicializar o contador de ticks
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- Para um contador zero, reinicie o buffer do indicador
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- Aumentar o contador
   ticks_counter++;
//--- Devemos periodicamente redefinir os contadores, para reviver a distribuição
   if(ticks_counter>100)
     {
      Print("Reajustamos os valores dos indicadores, vamos começar a preencher as célu
      ticks_counter=0;
     }
//--- Obter uma amostra de valores aleatórios como a soma de três números de 0 a 7
   for(int i=0;i<sample_number;i++)
     {
      //--- Cálculo do índice da célula, onde o número aleatório cai como a soma de ou
      int rand_index=0;
      //--- Obter três números aleatórios de 0 a 7
      for(int k=0;k<3;k++)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- Um restante na divisão por 7 retornará um valor de 0 a 6
         rand_index+=MathRand()%7;
        }
      //--- Aumentar o valor no número da célula rand_index por 1
      LabelBuffer[rand_index]++;
     }
//--- Sair do manipulador OnCalculate()
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

MathTan

A função retorna a tangente de um número.

```
double  MathTan(
   double  rad      // argumento em radianos
   );
```

Parâmetros

rad

[in] Ângulo em radianos.

Valor do Retorno

Tangente do rad. Se rad for maior ou igual a 263, ou menor ou igual a -263, uma perda de significância no resultado ocorre, neste caso a função retorna um número indefinido.

Observação

Ao invés de MathTan(), você pode usar tan().

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathIsValidNumber

Verifica a validade de um número real.

```
bool  MathIsValidNumber(
   double  number      // número para verificar
   );
```

Parâmetros

number

[in] Valor numérico verificado.

Valor do Retorno

Retorna true, se o valor verificado for um número real aceitável. Se o valor verificado for infinito positivo ou infinito negativo, ou "not a number" (NaN), a função retorna false.

Exemplo:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Atenção! MathArcsin(2.0) = ",abnormal);
```

Também Veja

Tipos Reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathExpm1

Retorna o valor da expressão MathExp(x)-1.

```
double  MathExpm1(
   double  value      // nível par a o número e
   );
```

Parâmetros

value

[in] Número que define o nível.

Valor de retorno

Número do tipo double. Após ser excedido, a função retorna INF (infinito), em caso de perda de precisão de MathExpm1, retorna 0.

Observação

Com valores *x* próximo de 0, a função MathExpm1(x) dá um valor muito mais preciso do que MathExp(x)-1.

Em vez da função MathExpm1(), é possível utilizar a função expm1().

Veja também

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathLog1p

Retorna o valor da expressão MathLog(1+x).

```
double  MathLog1p(
   double  value      // número para tomar o logaritmo
   );
```

Parâmetros

value

[in] Valor, cujo logaritmo deve ser calculado.

Valor de retorno

Logaritmo natural do valor (value+1), em caso de sucesso. Se value < -1, a função retorna NaN (valor não definido). Se value é igual a -1, a função retorna INF (infinito) .

Observação

Com valores x próximo de 0, a função MathLog1p(x) dá um valor muito mais preciso do que MathLog(1+x).

Em vez da função MathLog1p(), é possível utilizar a função log1p().

Veja também

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArccosh

Retorna o valor do arco cosseno hiperbólico.

```
double  MathArccosh(
   double  value     // 1 <= value < ∞
   );
```

Parâmetros

value

[in] Valor value, cujo arco-cosseno hiperbólico inverso deve ser calculado.

Valor de retorno

Arco-cosseno hiperbólico do número. Se value for inferior a +1, a função retornará NaN (valor não definido).

Observação

Em vez da função MathArccosh(), é possível utilizar a função acosh().

Veja também

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArcsinh

Retorna o valor do arco seno hiperbólico.

```
double  MathArcsinh(
   double  value      // -∞ < value < +∞
   );
```

Parâmetros

val

[in] Valor value, cujo arco-cosseno deve ser calculado.

Valor de retorno

Arco-seno hiperbólico de um número.

Observação

Em vez da função MathArcsinh(), é possível utilizar a função asinh().

Veja também

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathArctanh

Retorna o valor do arco tangente hiperbólico.

```
double  MathArctanh(
   double  value      // valor no intervalo -1 < value < 1
   );
```

Parâmetros

value

[in] Valor no intervalo -1 < value < 1, que representa a tangente.

Valor de retorno

Arco-tangente hiperbólica de um número.

Observação

Em vez da função MathArctanh(), é possível utilizar a função atanh().

© 2000-2024, MetaQuotes Ltd.

MathCosh

Retorna o valor do cosseno hiperbólico do número.

```
double  MathCosh(
   double  value      // número
   );
```

Parâmetros

value

[in] Valor.

Valor de retorno

Cosseno hiperbólico de um número, valor no intervalo de +1 a mais infinito.

Observação

Em vez da função MathCosh(), é possível utilizar a função cosh().

© 2000-2024, MetaQuotes Ltd.

MathSinh

Retorna o valor do seno hiperbólico do número.

```
double  MathSinh(
   double  value      // número
   );
```

Parâmetros

value

[in] Valor.

Valor de retorno

Seno hiperbólico de um número.

Observação

Em vez da função MathSinh(), é possível utilizar a função sinh().

© 2000-2024, MetaQuotes Ltd.

MathTanh

Retorna o valor da tangente hiperbólica числа.

```
double  MathTanh(
   double  value      // número
   );
```

Parâmetros

value

[in] Valor.

Valor de retorno

Tangente hiperbólica de um número, valor no intervalo de -1 a +1.

Observação

Em vez da função MathTanh(), é possível utilizar a função tanh().

Veja também

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

MathSwap

Altera a ordem de bytes em valores do tipo ushort.

```
ushort  MathSwap(
   ushort  value      // valor
   );
```

Parâmetros

value

[in] Valor para alterar a ordem de bytes.

Valor retornado

Valor ushort com ordem de bytes inversa.

MathSwap

Altera a ordem de bytes em valores do tipo uint.

```
uint  MathSwap(
   uint   value      // valor
   );
```

Parâmetros

value

[in] Valor para alterar a ordem de bytes.

Valor retornado

Valor uint com ordem de bytes inversa.

MathSwap

Altera a ordem de bytes em valores do tipo ulong.

```
ulong  MathSwap(
   ulong  value      // valor
   );
```

Parâmetros

value

[in] Valor para alterar a ordem de bytes.

Valor retornado

Valor ulong com ordem de bytes inversa.

Veja também

Funções de rede, SocketRead, SocketSend, SocketTlsRead, SocketTlsReadAvailable, SocketTlsSend

© 2000-2024, MetaQuotes Ltd.

Funções de String

Este é um grupo de funções destinado a trabalhar com dados do tipo string.

Função	Ação
StringAdd	Adiciona uma string no final de uma outra string
StringBufferLen	Retorna o tamanho de buffer alocado para uma string
StringCompare	Compara duas string e retorna 1 se a primeira string for maior que a segunda; 0 - se as string são iguais; -1 (menos 1) - se a primeira string for menor que a segunda.
StringConcatenate	Forma uma string dos parâmetros passados
StringFill	Preenche uma string especificada com símbolos selecionados
StringFind	Busca por uma substring em uma string
StringGetCharacter	Retorna o valor de um número localizado na posição de string especificada
StringInit	Inicializa uma string com símbolos especificados e fornece o comprimento da string especificada
StringLen	Retorna o número de símbolos em uma string
StringSetLength	Define para a string um comprimento especificado em caracteres
StringReplace	Substitui todas as substrings encontradas em uma string por uma seqüência de símbolos
StringReserve	Reserva na memória para uma string um buffer de tamanho especificado
StringSetCharacter	Retorna uma cópia de uma string com o valor alterado de um símbolo na posição especificada
StringSplit	Obtém substrings a partir de um separador especificado da string especificada, retorna o número de substrings obtidas
StringSubstr	Extrai uma substring a partir de uma string de texto começando a partir de uma posição especificada
StringToLower	Transforma todos os símbolos de uma string selecionada para minúscula pela localização
StringToUpper	Transforma todos os símbolos de uma string selecionada para maiúscula pela localização
StringTrimLeft	Remove caracteres de quebra de linha, espaços e tabs na parte esquerda da string
StringTrimRight	Remove caracteres de quebra de linha, espaços e tabs na parte direita da string

© 2000-2024, MetaQuotes Ltd.

StringAdd

A função adiciona uma substring no final de uma string.

```
bool  StringAdd(
   string&  string_var,        // string, na qual nós adicionados
   string   add_substring      // string, que é adicionada
   );
```

Parâmetros

string_var

[in][out] String, na qual uma outra string é adicionada.

add_substring

[in] String que é adicionada ao final de uma string fonte.

Valor do Retorno

Em caso de sucesso retorna verdadeiro, caso contrário falso. A fim de obter o código de erro, a função GetLastError() deve ser chamada.

Exemplo:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- primeiro método
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("tempo para 'c = a + b' = ",(stop-start)," milissegundos, i = ",i);

//--- segundo método
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("tempo para 'c = a + b' = ",(stop-start)," milissegundos, i = ",i);

//--- terceiro método
   start=GetTickCount();
   a="a"; // reinicializa a variável a
   for(i=0;i<length;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
    stop=GetTickCount();
    Print("tempo para 'StringConcatenate(c,a,b)' = ",(stop-start)," milissegundos, i =
  }
```

Também Veja

StringConcatenate, StringSplit, StringSubstr

© 2000-2024, MetaQuotes Ltd.

StringBufferLen

A função retorna o tamanho de buffer alocado para a string.

```
int  StringBufferLen(
   string  string_var      // string
   )
```

Parâmetros

string_var

[in] Seqüência.

Valor do Retorno

O valor 0 significa que a string é constante e o tamanho do buffer não pode ser alterado. -1 significa que a string pertence ao terminal cliente, e modificações no conteúdo do buffer pode ter resultados indeterminados.

Exemplo:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("antes: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("após: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Também Veja

StringAdd, StringInit, StringLen, StringFill

© 2000-2024, MetaQuotes Ltd.

StringCompare

A função compara duas strings e retorna o resultada da comparação na forma de um inteiro.

```
int  StringCompare(
   const string&  string1,                // a primeira string na comparação
   const string&  string2,                // a segunda string na comparação
   bool           case_sensitive=true     // seleção do modo sensibilidade à maiúscul
   );
```

Parâmetros

string1

[in] A primeira string.

string2

[in] A segunda string.

case_sensitive=true

[in] Seleção do modo de sensibilidade à maiúscula/minúscula. Se for true, então "A">"a". Se for false, então "A"="a". Por default, o valor é igual a true.

Valor do Retorno

- -1 (menos um), se string1<string2
- 0 (zero), se string1=string2
- 1 (um), se string1>string2

Observação

As strings são comparadas símbolo a símbolo, os símbolos são comparados na ordem alfabética em concordância como o código de página corrente.

Exemplo:

```
void OnStart()
  {
//--- qual é maior - apple ou home?
   string s1="Apple";
   string s2="home";

//--- compara com sensibilidade à maiúscula/minúscula
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Comparação sensível caso: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Comparação sensível caso: %s < %s",s1,s2);
      else PrintFormat("Comparação sensível caso: %s = %s",s1,s2);
     }

//--- Comparação insensível caso
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Comparação insensível caso: %s > %s",s1,s2);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       if(result2<0)PrintFormat("Comparação insensível caso: %s < %s",s1,s2);
       else PrintFormat("Comparação insensível caso: %s = %s",s1,s2);
      }
/* Resultado:
    Comparação sensível caso: Apple < home
    Comparação insensível caso: Apple < home
*/
  }
```

Também Veja

Tipo de String, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Uso de um Código de Página

© 2000-2024, MetaQuotes Ltd.

StringConcatenate

A função forma uma string de parâmetros passados e retorna o tamanho da string formada. Parâmetros podem ser de qualquer tipo. O número de parâmetros não pode ser menor que 2 ou maior que 64.

```
int  StringConcatenate(
   string&  string_var,    // string para formar
   void argument1          // primeiro parâmetro de qualquer tipo simples
   void argument2          // segundo parâmetro de qualquer tipo simples
   ...                     // próximo parâmetro de qualquer tipo simples
   );
```

Parâmetros

string_var

[out] String que será formada como resultado da concatenação.

argumentN

[in] Quaisquer valores separados por vírgula. De 2 a 63 parâmetros de qualquer tipo simples.

Valor do Retorno

Retorna o comprimento da string, formada pela concatenação dos parâmetros transformados em tipo string. Os parâmetros são transformados em strings em concordância com as mesmas regras aplicadas em Print() e Comment().

Também Veja

StringAdd, StringSplit, StringSubstr

© 2000-2024, MetaQuotes Ltd.

StringFill

Preenche uma string selecionada com um símbolo especificado.

```
bool  StringFill(
   string&   string_var,        // string a preencher
   ushort    character          // símbolos que preencherão a string
   );
```

Parâmetros

string_var

[in][out] String, que será preenchida pelo símbolo selecionado.

character

[in] Símbolo a partir do qual a string será preenchida.

Valor do Retorno

Em caso de sucesso retorna true, caso contrário - false. Para obter o código de erro chame GetLastError().

Observação

Preencher uma string no lugar significa que os símbolos são inseridos diretamente na string sem operações transitórias de criar ou copiar uma nova string. Isso permite economizar o tempo da operação.

Exemplo:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Resultado
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Também Veja

StringBufferLen, StringLen, StringInit

© 2000-2024, MetaQuotes Ltd.

StringFind

Busca por uma substring em uma string.

```
int  StringFind(
   string  string_value,        // string na qual a busca é feita
   string  match_substring,     // o que é procurado
   int     start_pos=0          // a partir de qual posição a busca inicia
   );
```

Parâmetros

string_value

[in] String, na qual a a busca é feita.

match_substring

[in] Substring procurada.

start_pos=0

[in] Posição na string a partir da qual a busca é iniciada.

Valor do Retorno

Retorna o número da posição na string, a partir da qual a substring procurada começa, ou -1, se a substring não foi encontrada.

Também Veja

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000-2024, MetaQuotes Ltd.

StringGetCharacter

Retorna o valor de um símbolo, localizado na posição especificada de uma string.

```
ushort  StringGetCharacter(
   string  string_value,      // string
   int     pos                // posição do símbolo na string
   );
```

Parâmetros

string_value

[in] Seqüência.

pos

[in] Posição de um símbolo na string. Pode ser de 0 a StringLen(text) -1.

Valor do Retorno

Código do símbolo ou 0 em caso de um erro. Para obter o código de erro chame GetLastError().

Também Veja

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

© 2000-2024, MetaQuotes Ltd.

StringInit

Inicializa uma string com um símbolo especificado e fornece o tamanho da string especificada.

```
bool  StringInit(
   string&   string_var,        // string para inicialização
   int       new_len=0,         // comprimento da string requerida após inicialização
   ushort    character=0        // símbolo, com o qual a string será preenchida
   );
```

Parâmetros

string_var

[in][out] String que deve ser inicializada e desinicializada.

new_len=0

[in] Comprimento da string após inicialização. Se length=0, ela desinicializa a string, isto é, o buffer da string é limpo e o endereço do buffer é zerado.

character=0

[in] Símbolo para preencher a string.

Valor do Retorno

Em caso de sucesso retorna true, caso contrário - false. Para obter o código de erro chame GetLastError().

Observação

Se character=0 e o comprimento new_len>0, o buffer da string de comprimento indicado será alocado e preenchido com zeros. O comprimento da string será igual a zero, porque o buffer inteiro será preenchido com terminadores de string.

Exemplo:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  Resultado:
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Também Veja

StringBufferLen, StringLen

© 2000-2024, MetaQuotes Ltd.

StringLen

Retorna o número de símbolos em uma string.

```
int  StringLen(
   string  string_value      // string
   );
```

Parâmetros

string_value

[in] String para calcular o comprimento.

Valor do Retorno

O número de símbolos em uma string sem o zero final.

Também Veja

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

© 2000-2024, MetaQuotes Ltd.

StringSetLength

Define para a string um comprimento especificado em caracteres.

```
bool  StringSetLength(
   string&    string_var,       // string
   uint       new_length        // novo comprimento de string
   );
```

Parâmetros

string_var

[in][out] String para a qual qual você deseja especificar um novo comprimento em caracteres.

new_capacity

[in] Comprimento necessário de string em caracteres. Se o novo tamanho new_length for menor do que o tamanho atual, os caracteres não selecionados serão descartados.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

A função StringSetLength() não altera o tamanho do buffer alocado para a string.

Veja também

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

© 2000-2024, MetaQuotes Ltd.

StringReplace

Substitui todas as substrings encontradas de uma string por uma seqüência de símbolos.

```
int  StringReplace(
   string&         str,              // a string na qual substrings serão substituídas
   const string    find,             // a substring procurada
   const string    replacement       // a substring que será inserida nas posições enc
   );
```

Parâmetros

str

[in][out] A string na qual você irá substituir substrings.

find

[in] A substring que deseja substituir.

replacement

[in] A string que será inserida na lugar da encontrada.

Valor do Retorno

A função retorna o número de substituições no caso de sucesso, caso contrário -1. Para obter o código do erro chame a função GetLastError().

Observação

Se a função rodou com sucesso, mas nenhuma substituição foi feita (a substring a ser substituída não foi encontrada), ela retorna 0.

O erro pode resultar de parâmetros *str* ou *find* incorretos (vazio ou string não inicializada, veja StringInit()). Além disso, o erro ocorre se não houver memória suficiente para concluir a substituição.

Exemplo:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  Resultado
//  Substituído: 3. Resultado=The slow black bear jumped over the lazy dog.
//
```

Também Veja

StringSetCharacter(), StringSubstr()

© 2000-2024, MetaQuotes Ltd.

StringReserve

Reserva na memória para uma string um buffer de tamanho especificado.

```
bool  StringReserve(
   string&    string_var,        // strings
   uint       new_capacity       // tamanho do buffer para armazenar a string.
   );
```

Parâmetros

string_var

[in][out] String para a qual você deseja alterar o tamanho do buffer.

new_capacity

[in] Tamanho do buffer necessário de acordo com a string. Se o novo tamanho new_capacity for menor que o comprimento da string, o tamanho do buffer atual não será alterado.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O tamanho da string não é igual ao tamanho do buffer para armazenar a string. Como regra, ao criar uma string, o buffer para ela é alocado com reserva. A função StringReserve() permite controlar o tamanho do buffer e especificar o tamanho ideal para operações futuras.

Ao contrário da StringInit(), a função StringReserve()) não altera o conteúdo da string e não a preenche com caracteres.

Exemplo:

```
void OnStart()
  {
   string s;
//--- verifica a velocidade sem usar StringReserve
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- agora mede usando a StringReserve
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- verifica o tempo
   Print("Test with StringReserve passed for "+(string)msc_reserve+" msc");
   Print("Test without StringReserve passed for "+(string)msc_no_reserve+" msc");
/* Resultado:
```

© 2000-2024, MetaQuotes Ltd.

```
     Test with StringReserve passed for 50 msc
     Test without StringReserve passed for 121 msc
*/
  }
```

Veja também

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

© 2000-2024, MetaQuotes Ltd.

StringSetCharacter

Retorna uma cópia de uma string com um caractere alterado na posição especificada.

```
bool  StringSetCharacter(
   string&   string_var,        // string
   int       pos,               // posição
   ushort    character          // caractere
   );
```

Parâmetros

string_var

[in][out] String.

pos

[in] Posição de um caractere em uma string. Pode ser de 0 a StringLen(text).

character

[in] Código Unicode de um símbolo.

Valor do Retorno

Em caso de sucesso retorna verdadeiro, caso contrário falso. A fim de obter o código de erro, a função GetLastError() deve ser chamada.

Observação

Se pos for menor que o comprimento da string e o valor de código do símbolo = 0, a string é truncada (mas o tamanho de buffer, alocado para a string permanece inalterado). O comprimento da string se torna igual a pos.

Se *pos* for igual ao comprimento da string, o símbolo especificado é adicionado ao final da string, e o comprimento é aumentado em um.

Exemplo:

```
void OnStart()
  {
   string str="0123456789";
   Print("antes: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- adicionado valor zero no meio
   StringSetCharacter(str,6,0);
   Print("após: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- adiciona símbolo no final
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("adição: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Resultado
```

© 2000-2024, MetaQuotes Ltd.

```
   antes: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    após: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
   adição: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Também Veja

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000-2024, MetaQuotes Ltd.

StringSplit

Obtém substrings por meio um separador especificado a partir de uma string especificada, retorna o número de substrings obtidas.

```
int  StringSplit(
   const string   string_value,        // A string que será pesquisada
   const ushort   separator,           // Um separador usado para buscar substrings
   string         & result[]           // Um array passado por referencia para obter as
   );
```

Parâmetros

string_value

[in] Uma string da qual você necessita obter substrings. A string não será alterada.

pos

[in] O código do caractere separador. Para obter o código, você pode usar a função StringGetCharacter().

result[]

[out] Um array de strings onde as substrings obtidas estão localizadas.

Valor do Retorno

O número de substrings no array result[]. Se o separador não for encontrado na string passada, somente uma string fonte será colocada no array.

Se string_value for vazia ou NULL, a função retornará zero. No caso de um erro, a função retorna -1. Para obter o código do erro, chame a função GetLastError().

Exemplo:

```
string to_split="_life_is_good_"; // Um string para dividir em substrings
   string sep="_";                // Um separador como um caractere
   ushort u_sep;                  // O código do caractere separador
   string result[];               // Um array para obter strings
   //--- Obtém o código do separador
   u_sep=StringGetCharacter(sep,0);
   //--- Divide a string em substrings
   int k=StringSplit(to_split,u_sep,result);
   //--- Exibe um comentário
   PrintFormat("Strings obtidos: %d. Usado separador '%s' com o código %d",k,sep,u_sep
   //--- Agora imprime todos os resultados obtidos
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

© 2000-2024, MetaQuotes Ltd.

Também Veja

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2024, MetaQuotes Ltd.

StringSubstr

Extrai uma substring de uma string de texto iniciando de uma posição especificada.

```
string  StringSubstr(
   string  string_value,      // string
   int     start_pos,         // posição de início
   int     length=-1          // comprimento da string extraída
   );
```

Parâmetros

string_value

[in] String de onde será extraída uma substring.

start_pos

[in] Posição inicial de uma substring. Pode ser de 0 a StringLen(text) -1.

length=-1

[in] Comprimento de uma substring extraída. Se o valor do parâmetro for igual a -1 ou o parâmetro não for definido, a substring será extraída da posição indicada até a final da string.

Valor do Retorno

Cópia de uma substring extraída, se possível. Caso contrário, retorna uma string vazia.

Também Veja

StringSplit, StringFind, StringGetCharacter

© 2000-2024, MetaQuotes Ltd.

StringToLower

Transforma todos os símbolos de uma string selecionada em minúscula por localização.

```
bool  StringToLower(
   string&  string_var      // string para processar
   );
```

Parâmetros

string_var

[in][out] String.

Valor do Retorno

Em caso de sucesso retorna true, caso contrário - false. Para obter o código de erro chame GetLastError().

Também Veja

StringToUpper, StringTrimLeft, StringTrimRight

© 2000-2024, MetaQuotes Ltd.

StringToUpper

Transforma todos os símbolos de uma string selecionada em maiúscula por localização.

```
bool  StringToUpper(
   string&  string_var      // string para processar
   );
```

Parâmetros

string_var

[in][out] String.

Valor do Retorno

Em caso de sucesso retorna true, caso contrário - false. Para obter o código de erro chame GetLastError().

Também Veja

StringToLower, StringTrimLeft, StringTrimRight

© 2000-2024, MetaQuotes Ltd.

StringTrimLeft

A função remove caracteres de quebra de linha, espaços e tabs na parte esquerda da string até o primeiro símbolo significativo. A string é modificada no lugar.

```
int  StringTrimLeft(
   string&  string_var      // string para cortar
   );
```

Parâmetros

string_var

[in][out] String que será cortada a partir da esquerda.

Valor do Retorno

Retorna o número de símbolos removidos.

Também Veja

StringTrimRight, StringToLower, StringToUpper

© 2000-2024, MetaQuotes Ltd.

StringTrimRight

A função remove caracteres de quebra de linha, espaços e tabs na parte direita da string após o último símbolo significativo. A string é modificada no lugar.

```
int  StringTrimRight(
   string&  string_var      // string para cortar
   );
```

Parâmetros

string_var

[in][out] String que será cortada a partir da direita.

Valor do Retorno

Retorna o número de símbolos removidos.

Também Veja

StringTrimLeft, StringToLower, StringToUpper

© 2000-2024, MetaQuotes Ltd.

Data e Hora

Este é o grupo de funções para trabalhar com dados do tipo datetime (um inteiro que representa número de segundos decorridos a partir da 0 hora de 1° Janeiro de 1970).

Para obter contadores e timers de alta-resolução, use a função GetTickCount(), que produz valores em milissegundos.

Função	Ação
TimeCurrent	Retorna a última hora conhecida do servidor (hora da última cotação recebida) no formato de datetime(
TimeTradeServer	Retorna a hora calculada corrente do servidor de negociação
TimeLocal	Retorna a hora do computador local em formato datetime
TimeGMT	Retorna a hora GMT em formato datetime com o horário de verão por hora local do computador, de onde o terminal cliente está executando
TimeDaylightSavings	Retorna o sinal do ajuste do horário de verão
TimeGMTOffset	Retorna a diferença corrente entre a hora GMT e a hora do computador local em segundos, levando em consideração o ajuste do horário de verão
TimeToStruct	Converte um valor datetime em uma variável do tipo estrutura MqlDateTime
StructToTime	Converte uma variável do tipo estrutura MqlDateTime em um valor datetime

© 2000-2024, MetaQuotes Ltd.

TimeCurrent

Retorna a última hora conhecida do servidor, hora da última cotação recebida para um dos ativos selecionados na janela "Observação de Mercado". No handler OnTick(), esta função retorna a hora do tick recebido, que está sendo tratado. Em outros casos (por exemplos, chamar nos handlers OnInit(), OnDeinit(), OnTimer() e assim por diante), esta função retorna a hora da última cotação recebida para qualquer ativo disponível na janela "Observação de Mercado", a hora exibida no cabeçalho desta janela. O valor da hora é formado em servidor de negociação e não depende das configurações de hora em seu computador. Existem 2 variantes da função.

Chamar sem parâmetros

```
datetime  TimeCurrent();
```

Chamar com parâmetro de tipo MqlDateTime

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // variável de tipo estrutura
   );
```

Parâmetros

dt_struct

[out] MqlDateTime variável de tipo estrutura.

Valor do Retorno

Valor de tipo datetime

Observação

Se uma variável de tipo estrutura MqlDateTime for passada como parâmetro, ela é preenchida de acordo.

Para obter contadores e timer de alta-resolução, use a função GetTickCount(), que produz valores em milissegundos.

During testing in the strategy tester, TimeCurrent() is simulated according to historical data.

© 2000-2024, MetaQuotes Ltd.

TimeTradeServer

Returna a hora corrente calculada do servidor de negociação. Diferentemente de TimeCurrent(), o cálculo do valor da hora é realizado no terminal cliente e depende das configurações de hora em seu computador. Existem 2 variantes da função.

Chamar sem parâmetros

```
datetime  TimeTradeServer();
```

Chamar com parâmetro de tipo MqlDateTime

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // Variável de tipo estrutura
   );
```

Parâmetros

dt_struct

[out] Variável de tipo estrutura MqlDateTime.

Valor do Retorno

Valor de tipo datetime

Observação

Se uma variável de tipo estrutura MqlDateTime for passada como parâmetro, ela é preenchida de acordo.

Para obter contadores e timer de alta-resolução, use a função GetTickCount(), que produz valores em milissegundos.

Durante o teste no testador de estratégia, o tempo do servidor simuladoTimeGMT() é sempre igual ao TimeCurrent().

© 2000-2024, MetaQuotes Ltd.

TimeLocal

Retorna a hora local de um computador, de onde o terminal cliente está executando. Existem 2 variantes da função.

Chamar sem parâmetros

```
datetime  TimeLocal();
```

Chamar com parâmetro de tipo MqlDateTime

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // Variável de tipo estrutura
   );
```

Parâmetros

dt_struct

[out] Variável de tipo estrutura MqlDateTime.

Valor do Retorno

Valor de tipo datetime

Observação

Se uma variável de tipo estrutura MqlDateTime for passada como parâmetro, ela é preenchida de acordo.

Para obter contadores e timer de alta-resolução, use a função GetTickCount(), que produz valores em milissegundos.

Durante o teste no testador de estratégia, o tempo do servidor simuladoTimeGMT() é sempre igual ao TimeCurrent().

© 2000-2024, MetaQuotes Ltd.

TimeGMT

Retorna a hora GMT, que é calculada levando em consideração o ajuste ao horário de verão pela hora local no computador de onde o terminal cliente está executando. Existem 2 variantes da função.

Chamar sem parâmetros

```
datetime  TimeGMT();
```

Chamar com parâmetro de tipo MqlDateTime

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // Variável de tipo estrutura
   );
```

Parâmetros

dt_struct

[out] Variável de tipo estrutura MqlDateTime.

Valor do Retorno

Valor de tipo datetime

Observação

Se uma variável de tipo estrutura MqlDateTime for passada como parâmetro, ela é preenchida de acordo.

Para obter contadores e timer de alta-resolução, use a função GetTickCount(), que produz valores em milissegundos.

Durante o teste no testador de estratégia, o tempo do servidor simuladoTimeGMT() é sempre igual ao TimeTradeServer().

© 2000-2024, MetaQuotes Ltd.

TimeDaylightSavings

Retorna correção para o horário de verão em segundos, ajuste ao horário de verão foi feito. Depende das configurações de hora de seu computador.

```
int  TimeDaylightSavings();
```

Valor do Retorno

Se o ajuste ao horário de inverno (padrão) foi feito, retorna 0.

© 2000-2024, MetaQuotes Ltd.

TimeGMTOffset

Retorna a diferença corrente entre a hora GMT e a hora do computador local em segundos. Depende das configurações de hora de seu computador.

```
int  TimeGMTOffset();
```

Valor do Retorno

Valor de tipo int, representando a diferença corrente entre a hora GMT e a hora local do computador TimeLocal() em segundos.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

© 2000-2024, MetaQuotes Ltd.

TimeToStruct

Converte um valor de tipo datetime (número de segundos desde 01.01.1970) em uma variável estrutura MqlDateTime.

```
bool  TimeToStruct(
   datetime      dt,            // data e hora
   MqlDateTime&  dt_struct      // estrutura para a captura de valores
   );
```

Parâmetros

dt

[in] Valor de data e hora para conversão.

dt_struct

[out] Variável de tipo estrutura MqlDateTime.

Valor do Retorno

true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

StructToTime

Converte uma variável estrutura MqlDateTime em um valor de tipo datetime e retorna o valor resultante.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // estrutura da data e hora
   );
```

Parâmetros

dt_struct

[in] Variável de tipo estrutura MqlDateTime.

Valor do Retorno

Valor de tipo datetime contendo o número de segundos deste 01.01.1970.

© 2000-2024, MetaQuotes Ltd.

Informações de Conta

Funções que retornam parâmetros da conta corrente.

Função	Ação
AccountInfoDouble	Retorna um valor de tipo double da correspondente propriedade da conta
AccountInfoInteger	Retorna um valor de tipo inteiro (bool, int ou long) da correspondente propriedade da conta
AccountInfoString	Retorna um valor de tipo string da correspondente propriedade da conta

© 2000-2024, MetaQuotes Ltd.

AccountInfoDouble

Retorna o valor da correspondente propriedade da conta.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade. O valor pode ser um dos valores de ENUM_ACCOUNT_INFO_DOUBLE.

Valor do Retorno

Value of double type.

Exemplo:

```
void OnStart()
  {
//--- exibe todas as informações disponíveis a partir da função AccountInfoDouble()
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

Também Veja

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2024, MetaQuotes Ltd.

AccountInfoInteger

Retorna o valor das propriedades da conta.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // Identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade. O valor pode ser um dos valores de ENUM_ACCOUNT_INFO_INTEGER.

Valor do Retorno

Value of long type.

Observação

A propriedade deve ser um dos tipos bool, int ou long.

Exemplo:

```
void OnStart()
  {
//--- exibe todas as informações disponíveis a partir da função AccountInfoInteger()
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoIntegei

//--- Informa sobre a possibilidade de realizar uma operação de negociação
   if(thisAccountTradeAllowed)
      Print("Comércio para esta conta é permitido");
   else
      Print("Comércio para esta conta é proibido!");

//--- Descobre se é possível negociar neste conta através de Expert Advisors
   if(EATradeAllowed)
      Print("Comércio com Expert Advisors é permitido para esta conta");
   else
      Print("Comércio com Expert Advisors é proibido para esta conta!");

//--- Descobre o tipo de conta
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("Esta é uma conta demonstração");
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("Esta é uma conta de competição");
         break;
      default:Print("Esta é uma conta real!");
     }

//--- Descobre o modo de configuração de nível StopOut
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("O nível StopOut é especificado em porcentagem");
         break;
      default:Print("O nível StopOut é especificado monetariamente");
     }
  }
```

Também Veja

Informações de Conta

© 2000-2024, MetaQuotes Ltd.

AccountInfoString

Retorna o valor da correspondente propriedade da conta.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // Identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade. O valor pode ser um dos valores de ENUM_ACCOUNT_INFO_STRING.

Valor do Retorno

Valor de tipo string.

Exemplo:

```
void OnStart()
  {
//--- Exibe todas as informações disponíveis a partir da função AccountInfoString()
   Print("O nome do corretor = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Moeda do depósito = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Nome do cliente = ",AccountInfoString(ACCOUNT_NAME));
   Print("O nome do servidor comercial = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Também Veja

Informações de Conta

© 2000-2024, MetaQuotes Ltd.

Verificação de Estado

Funções que retornam parâmetros do estado corrente do terminal cliente

Função	Ação
GetLastError	Retorna o último erro
IsStopped	Retorna true, se um programa mql5 foi ordenado a interromper sua operação
UninitializeReason	Retorna o código do motivo da desinicialização
TerminalInfoInteger	Retorna um valor inteiro de uma propriedade correspondente do ambiente de um programa mql5
TerminalInfoDouble	Retorna um valor de double de uma propriedade correspondente do ambiente de um programa mql5
TerminalInfoString	Retorna um valor de string de uma propriedade correspondente do ambiente de um programa mql5
MQLInfoInteger	Retorna um valor inteiro de uma propriedade correspondente de um programa mql5 em execução
MQLInfoString	Retorna um valor de string de uma propriedade correspondente de um programa mql5 em execução
Symbol	Retorna o nome do ativo do gráfico corrente
Period	Retorna a janela de tempo do gráfico corrente
Digits	Retorna o número de dígitos decimais determinando a precisão do valor do preço do ativo do gráfico corrente
Point	Retorna o tamanho do ponto do símbolo atual na moeda de cotação.

© 2000-2024, MetaQuotes Ltd.

GetLastError

Retorna o conteúdo da variável de sistema _LastError.

```
int  GetLastError();
```

Valor de Retorno

Retorna o valor do último erro que ocorreu durante a execução de um programa mql5.

Observação

Após a chamada da função, o conteúdo de _LastError não é zerado. Para zerar esta variável, você precisa chamar ResetLastError().

Também Veja

Códigos de Retorno do Servidor de Negociação

© 2000-2024, MetaQuotes Ltd.

IsStopped

Verifique um desligamento forçado de um programa mql5.

```
bool  IsStopped();
```

Valor de Retorno

Retorna true, se a variável de sistema _StopFlag contém um valor diferente de 0. Um valor não-zero é escrito em _StopFlag, se um programa mql5 foi ordenado a concluir sua operação. Neste caso, você deve imediatamente finalizar o programa, caso contrário o programa será concluído forçadamente do lado de fora após 3 segundos.

© 2000-2024, MetaQuotes Ltd.

UninitializeReason

Retorna o código do motivo para desinicialização.

```
int  UninitializeReason();
```

Valor de Retorno

Retorna o valor de _UninitReason que é formado antes de OnDeinit() ser chamado. O valor depende dos motivos que levaram à desinicialização.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoInteger

Retorna o valor de uma propriedade correspondente do ambiente de um programa mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Can be one of the values of the enumeration ENUM_TERMINAL_INFO_INTEGER.

Valor de Retorno

Valor de tipo int.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoDouble

Retorna o valor de uma propriedade correspondente do ambiente do programa mql5.

```
double  TerminalInfoDouble(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Pode ser um dos valores da enumeração ENUM_TERMINAL_INFO_DOUBLE.

Retornar Valor

Valor do tipo double.

© 2000-2024, MetaQuotes Ltd.

TerminalInfoString

A função retorna o valor de uma propriedade correspondente do ambiente de um programa mql5. A propriedade deve ser do tipo string.

```
string  TerminalInfoString(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Pode ser um dos valores da enumeração ENUM_TERMINAL_INFO_STRING.

Valor de Retorno

Valor de tipo string.

© 2000-2024, MetaQuotes Ltd.

MQLInfoInteger

Retorna o valor de uma propriedade correspondente de um programa mql5 em execução.

```
int  MQLInfoInteger(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Can be one of values of the ENUM_MQL_INFO_INTEGER enumeration.

Valor de Retorno

Valor de tipo int.

© 2000-2024, MetaQuotes Ltd.

MQLInfoString

Retorna o valor de uma propriedade correspondente de um programa mql5 em execução.

```
string  MQLInfoString(
   int  property_id      // Identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Pode ser um dos valores da enumeração ENUM_MQL_INFO_STRING.

Valor de Retorno

Valor de tipo string.

© 2000-2024, MetaQuotes Ltd.

Symbol

Retorna o nome do ativo do gráfico corrente.

```
string  Symbol();
```

Valor de Retorno

Valor da variável de sistema _Symbol, que armazena o nome do ativo do gráfico corrente.

Observação

Ao contrário de EAs, indicadores e scripts, os serviços não estão vinculados a um gráfico específico. Portanto, para o serviço, Symbol() retorna a linha em branco ("").

© 2000-2024, MetaQuotes Ltd.

Period

Retorna a janela de tempo do gráfico corrente.

```
ENUM_TIMEFRAMES  Period();
```

Valor de Retorno

O conteúdo da variável _Period que representa o valor da janela de tempo do gráfico corrente. O valor pode ser um dos valores da enumeração ENUM_TIMEFRAMES.

Observação

Ao contrário de EAs, indicadores e scripts, os serviços não estão vinculados a um gráfico específico. Portanto, para o serviço, Period() retorna 0.

Veja também

PeriodSeconds, Janelas de tempo de gráficos, Data e Hora, Visibilidade de objetos

© 2000-2024, MetaQuotes Ltd.

Digits

Retorna o número de dígitos decimais determinando a precisão do preço do ativo do gráfico corrente.

```
int  Digits();
```

Valor de Retorno

O valor da variável _Digits que armazena o número de dígitos decimais determina a precisão do preço do ativo do gráfico corrente.

© 2000-2024, MetaQuotes Ltd.

Point

Retorna o tamanho do ponto do ativo corrente na moeda de cotação.

```
double  Point();
```

Valor de Retorno

O valor da variável _Point que armazena o tamanho do ponto do ativo corrente na moeda de cotação.

© 2000-2024, MetaQuotes Ltd.

Manipulação de eventos

A linguagem MQL5 providencia o processamento de eventos predefinidos. As funções para processar esses eventos devem ser definidas no programa MQL5: o nome da função, o tipo de retorno, a composição dos parâmetros (se houver) e seus tipos devem corresponder estritamente à descrição da função do manipulador de eventos.

É pelo tipo do valor de retorno e pelos tipos de parâmetros que o manipulador de eventos do terminal do cliente identifica as funções que manipulam eventos. Se, para a respectiva função, forem definidos outros parâmetros que não correspondem às seguintes descrições, ou se for especificado um tipo de retorno diferente, essa função não será usada para processar o evento.

Função	Descrição
OnStart	É chamada no script quando ocorre um evento Start para executar ações definidas no script
OnInit	É chamada em indicadores e EAs quando ocorre um evento Init para inicializar um programa MQL5 em execução
OnDeinit	É chamada em indicadores e EAs quando ocorre um evento Deinit para desinicializar um programa MQL5 em execução
OnTick	É chamada em EAs quando ocorre um evento NewTick para processar uma nova cotação
OnCalculate	É chamada em indicadores quando ocorre um evento Calculate para processar alterações de dados de preço
OnTimer	É chamada em indicadores e EAs no início de um evento periódico Timer que é gerado pelo terminal com o intervalo de tempo definido
OnTrade	É chamada em EAs quando ocorre um evento Trade que é gerado no final da negociação no servidor de negociação
OnTradeTransaction	É chamada em EAs quando ocorre um evento TradeTransaction para processar os resultados de uma solicitação de negociação
OnBookEvent	É chamada em EAs quando ocorre um evento BookEvent para processar alterações no livro de ofertas
OnChartEvent	É chamada em indicadores e Eas quando ocorre um evento ChartEvent para processar mudanças no gráfico causadas por ações do usuário ou pelo funcionamento de programas MQL5
OnTester	É chamada em EAs quando ocorre um evento Tester para executar as ações necessárias no final do teste
OnTesterInit	É chamada em EAs quando ocorre um evento TesterInit para executar as ações necessárias antes do início da otimização
OnTesterDeinit	É chamada em EAs quando ocorre um evento TesterDeinit para realizar as ações necessárias após a otimização do EA

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
OnTesterPass	É chamada em EAs quando ocorre um evento TesterPass para processar a chegada de um novo quadro de dados durante a otimização do EA

O terminal do cliente envia os novos eventos para os respectivos gráficos abertos. Além disso, os eventos podem ser gerados por gráficos (evento de gráfico) ou por programas MQL5 (eventos personalizados). A geração de eventos para criar e excluir objetos gráficos no gráfico pode ser ativada e desativada definindo as propriedades do gráfico CHART_EVENT_OBJECT_CREATE e CHART_EVENT_OBJECT_DELETE. Cada programa MQL5 e cada gráfico tem sua própria fila de eventos, à qual são adicionados todos os eventos recém-chegados.

O programa recebe eventos apenas do gráfico em que está sendo executado. Todos os eventos são processados um após o outro na ordem em que são recebidos. Se na fila já houver um evento NewTick ou este evento estiver no estado de processamento, o novo evento NewTick não será colocado na fila do programa MQL5. Da mesma forma, se na fila do programa MQL5 já estiver o evento ChartEvent ou tal evento estiver sendo processado, não será enfileirado um novo evento desse tipo. Os eventos do temporizador são processados da mesma maneira, isto é, se um evento estiver na fila ou o evento Timer já estiver sendo processado, não será enfileirado um novo evento do temporizador.

Filas de eventos são de tamanho limitado, mas suficiente, portanto, para um programa escrito corretamente, o estouro da fila é improvável. Quando a fila estourar, novos eventos são descartados sem serem enfileirados.

Evite usar loops infinitos para manipular eventos. Uma exceção a essa regra podem ser apenas os scripts que lidam com o evento único Start.

As bibliotecas não processam nenhum evento.

© 2000-2024, MetaQuotes Ltd.

OnStart

É chamado em scripts e serviços quando ocorre um evento Start. A função se destina à execução única de ações estabelecidas pelo programa. Existem duas versões da função.

Versão com retorno de resultado

```
int  OnStart(void);
```

Valor retornado

Valor de tipo int que é exibido na guia "Diário".

Após o script ser concluído, no log do terminal será criada uma entrada de tipo "script nome_do_script removed (result code N)", onde N é o valor retornado pela função OnStart().

Após o serviço ser concluído, no log do terminal será criada uma entrada de tipo "service имя_сервиса stopped (result code N)", onde N é o valor retornado pela função OnStart().

O uso da chamada da OnStart() com o retorno de resultado é prioridade, pois este método permite não apenas iniciar o script, mas também retornar um código de erro ou outras informações úteis para analisar o resultado da execução do script.

Versão sem retorno do resultado é deixada apenas para compatibilidade com códigos antigos. Não é recomendada

```
void  OnStart(void);
```

Observação

OnStart() é a única função para manipular eventos em scripts e serviços, outros eventos não são enviados para esses programas. Por sua vez, o evento Start não é enviado para EAs e indicadores personalizados.

Exemplo de script:

```
//--- macros para trabalhar com cores
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- definimos a cor a vela de baixa
   Comment("Definimos a cor da vela de baixa");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // atualizamos imediatamente o gráfico sem esperar por um novo tick
   Sleep(1000);   // fazemos uma pausa de 1 segundo, para poder ver as alterações
//--- definimos a cor da vela de alta
   Comment("Definimos a cor da vela de alta");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definimos a cor de fundo
   Comment("Definimos a cor de fundo");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da linha Ask
   Comment("Definimos a cor da linha Ask");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da linha Bid
   Comment("Definimos a cor da linha Bid");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da barra baixista e as bordas da vela baixista
   Comment("Definimos a cor da barra baixista e as bordas da vela baixista");
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da linha do gráfico e das velas do tipo "Doji"
   Comment("Definimos a cor da linha do gráfico e das velas do tipo Doji");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da barra altista e a borda da vela altista
   Comment("Definimos a cor da barra altista e a borda da vela altista");
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor dos eixos, da escala e da linha OHLC
   Comment("Definimos a cor dos eixos, da escala e da linha OHLC");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor da grade
   Comment("Definimos a cor da grade");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor do preço Last
   Comment("Definimos a cor do preço Last");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- definimos a cor dos níveis das ordens Stop Loss e Take Profit
   Comment("Definimos a cor dos níveis das ordens Stop Loss e Take Profit");
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
```

© 2000-2024, MetaQuotes Ltd.

```
   Sleep(1000);
//--- definimos a cor dos volumes e dos níveis de abertura das posições
   Comment("Definimos a cor dos volumes e dos níveis de abertura das posições");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Retorna uma cor gerada aleatoriamente                            |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Veja também

Funções de processamento de eventos, Execução de programas, Eventos do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnInit

É chamada em indicadores e EAs quando ocorre o evento Init. A função é projetada para inicialização de um programa MQL5 em execução. Existem duas versões da função.

Versão com retorno de resultado

```
int  OnInit(void);
```

Valor retornado

Valor de tipo int, zero significa inicialização bem-sucedida.

Ao retornar o valor INIT_FAILED o Expert Advisor será retirado à força do gráfico.

Ao retornar o valor INIT_FAILED o indicador não será removido do gráfico. Nesse caso, o indicador restante no gráfico estará inoperante — os manipuladores de eventos não são chamados no indicador.

O uso da chamada da OnInit() com o retorno de resultado é prioridade, pois este método permite não apenas inicializar o programa, mas também retornar o código de erro em caso de término antecipado do programa.

Versão sem retorno do resultado é deixada apenas para compatibilidade com códigos antigos. Não é recomendada

```
void  OnInit(void);
```

Observação

O evento Init é gerado imediatamente após o especialista ou indicador ser carregado, este evento não é gerado para scripts. A função OnInit() é usada para inicialização do programa MQL5. Se a OnInit() tiver um valor retornado do tipo int, um código de retorno diferente de zero significa uma inicialização mal-sucedida e gera o evento Deinit com código de motivo de desinicialização REASON_INITFAILED.

A função OnInit() do tipo void sempre significa uma inicialização bem-sucedida e não é recomendada.

Ao otimizar os parâmetros de entrada do EA, recomenda-se usar como código de retorno os valores da enumeração ENUM_INIT_RETCODE. Estes valores se destinam a organizar o gerenciamento do progresso da otimização e, inclusive, a selecionar os mais adequados agentes de teste. Na inicialização do EA, mesmo antes de executar o teste, você pode solicitar informações sobre a configuração e sobre os recursos do agente (o número de núcleos, a quantidade de memória livre, etc.) usando a função TerminalInfoInteger(). Com base nas informações recebidas, você pode permitir que esse agente de teste seja usado ou não ao ser otimizado.

Identificador	Descrição
INIT_SUCCEEDED	Inicialização bem-sucedida, o teste do EA pode continuar. Esse código significa o mesmo que o valor zero - inicialização do EA no testador bem-sucedida.
INIT_FAILED	Inicialização mal-sucedida, não faz sentido continuar o teste devido a erros irrecuperáveis. Por exemplo, não foi possível

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	criar o indicador necessário para o trabalho do EA. O retorno deste valor significa o mesmo que o retorno, retrocesso de um valor diferente de zero - inicialização do EA no testador mal-sucedida.
INIT_PARAMETERS_INCORRECT	Projetado para denotar um conjunto incorreto de parâmetros de entrada, na tabela de otimização geral, a sequência de resultados com esse código de retorno será destacada em vermelho. O teste para este conjunto de parâmetros do EA não será executado, o agente estará livre para receber um novo trabalho. Quando esse valor é recebido, o testador de estratégia tem a garantia de não transferir essa tarefa para outros agentes para reexecução.
INIT_AGENT_NOT_SUITABLE	Não há erros no programa durante a inicialização, mas por algum motivo este agente não é adequado para testes. Por exemplo, não há RAM suficiente, sem suporte de OpenCL e assim por diante. Após retornar este código, o agente não receberá mais tarefas até o final desta otimização.

O uso da OnInit() com retorno de INIT_FAILED/INIT_PARAMETERS_INCORRECT no testador tem várias características que devem ser consideradas ao otimizar EAs:

- o conjunto de parâmetros para o qual a OnInit() retornou INIT_PARAMETERS_INCORRECT é considerado inadequado para testes e não será usado para obter a próxima população duranteotimizações genéticas. O fato de haver muitos conjuntos de parâmetros rejeitados pode levar a resultados de pesquisa incorretos de parâmetros ideais de EAs. O algoritmo de busca pressupõe que a função de critério de otimização é suave e não tem descontinuidades em todo o conjunto de parâmetros de entrada.
- se a OnInit() retornar INIT_FAILED, o teste não poderá ser iniciado e o EA será descarregado da memória do agente. Para executar a próxima passagem, o novo EA será carregado com o novo conjunto de parâmetros. Isso faz com que inciar o teste da próxima passagem de otimização leve mais tempo do que se fosse chamada a TesterStop().

Exemplo da função OnInit() para um EA

```
//--- input parameters
input int      ma_period=20; // período de média móvel

//--- identificador do indicador usado no Expert Advisor
int indicator_handle;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificamos se o valor ma_period é correto
```

© 2000-2024, MetaQuotes Ltd.

```
   if(ma_period<=0)
     {
      PrintFormat("Valor inválido do parâmetro de entrada ma_period: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- ao otimizar
   if(MQLInfoInteger(MQL_OPTIMIZATION))
     {
      //--- verificamos a quantidade de RAM disponível para o agente
      int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
      if(available_memory_mb<2000)
        {
         PrintFormat("Memória insuficiente para o agente de teste: %d MB",
                     available_memory_mb);
         return (INIT_AGENT_NOT_SUITABLE);
        }
     }
//--- verificamos se existe indicador
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("Não foi possível criar o indicador My_Indicator. Código de erro %d"
                  GetLastError());
      return (INIT_FAILED);
     }
//--- inicialização do EA bem-sucedida
   return(INIT_SUCCEEDED);
  }
```

Veja também

OnDeinit, TesterHideIndicators, Funções de processamento de eventos, Execução de programas, Evento do terminal do cliente, Inicialização de variáveis, Criando e destruindo objetos

© 2000-2024, MetaQuotes Ltd.

OnDeinit

É chamada em indicadores e EAs quando ocorre o evento Deinit. A função é projetada para desinicialização de um programa MQL5 em execução.

```
void  OnDeinit(
   const int  reason         // código de motivo da desinicialização
   );
```

Parâmetros

reason

[in] Código de motivo da desinicialização.

Valor retornado

Sem valor retornado

Observação

O evento Deinit é gerado para EAs e para indicadores nos seguintes casos:

- antes de uma reinicialização devida à mudança do símbolo ou do período do gráfico ao qual o programa MQL5 está associado;
- antes de uma reinicialização devida à mudança dos parâmetros de entrada;
- antes de descarregar o programa MQL%.

O parâmetro *reason* pode pegar os seguintes valores:

Constante	Valor	Descrição
REASON_PROGRAM	0	O Expert Advisor parou seu trabalho chamando a função ExpertRemove()
REASON_REMOVE	1	Programa excluído do gráfico
REASON_RECOMPILE	2	Programa recompilado
REASON_CHARTCHANGE	3	Símbolo ou período alterado
REASON_CHARTCLOSE	4	Gráfico fechado
REASON_PARAMETERS	5	Parâmetros de entrada alterados pelo usuário
REASON_ACCOUNT	6	Outra conta ativada ou reconectada ao servidor de negociação como resultado da alteração das configurações da conta
REASON_TEMPLATE	7	Outro modelo de gráfico implementado
REASON_INITFAILED	8	Manipulador de OnInit() retornou um valor diferente de zero
REASON_CLOSE	9	Terminal fechado

Códigos de motivo de desinicialização do EA podem ser obtidos pela função UninitializeReason() ou a partir da variável predefinida _UninitReason.

© 2000-2024, MetaQuotes Ltd.

Exemplo da funções OnInit() e OnDeinit() para o EA

```
input int fake_parameter=3;      // parâmetro inútil
//+----------´-----------------------------------------------------------+
//| Expert initialization function                                       |
//+----------´-----------------------------------------------------------+
int OnInit()
  {
//--- Obtemos o número da compilação do programa
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- Código de motivo de reinicialização pode ser obtido na OnInit()
   Print(__FUNCTION__," Ao reinicializar o EA, você pode obter o código de motivo da c
//--- Primeira maneira de obter o código de motivo de desinicialização
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Segunda maneira de obter o código de motivo de desinicialização
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+----------´-----------------------------------------------------------+
//| Expert deinitialization function                                     |
//+----------´-----------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Primeira maneira de obter o código de motivo de desinicialização
   Print(__FUNCTION__," Código de motivo da desinicialização = ",reason);
//--- Segunda maneira de obter o código de motivo de desinicialização
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Terceira maneira de obter o código de motivo de desinicialização
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| Retorna a descrição do motivo da desinicialização                |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Account was changed";break;
      case REASON_CHARTCHANGE:
         text="Symbol or timeframe was changed";break;
      case REASON_CHARTCLOSE:
         text="Chart was closed";break;
      case REASON_PARAMETERS:
         text="Input-parameter was changed";break;
      case REASON_RECOMPILE:
```

© 2000-2024, MetaQuotes Ltd.

```
         text="Program "+__FILE__+" was recompiled";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" was removed from chart";break;
      case REASON_TEMPLATE:
         text="New template was applied to chart";break;
      default:text="Another reason";
     }
//---
   return text;
  }
```

Veja também

OnInit, Funções de processamento de eventos, Execução de programas, Eventos do terminal do cliente, Motivos de desinicialização, Escopo de visibilidade e tempo de vida das variáveis, Criando e destruindo objetos

© 2000-2024, MetaQuotes Ltd.

OnTick

É chamada em EAs quando ocorre o evento NewTick para processar uma nova cotação.

```
void  OnTick(void);
```

Valor retornado

Sem valor retornado

Observação

O evento NewTick é gerado apenas para EAs após a chegada de um novo tick do símbolo cujo EA é anexado ao gráfico. É inútil definir a função OnTick() no indicador personalizado ou no script, pois o evento NewTick não é gerado para eles.

O evento Tick é gerado apenas para EAs, mas isso não significa que os EAs precisam ter a função OnTick(), porque não apenas os eventos NewTick são gerados para EAs, mas também os eventos Timer, BookEvent e ChartEvent.

Todos os eventos são processados um após o outro na ordem em que são recebidos. Se na fila já houver um evento NewTick ou este evento estiver no estado de processamento, o novo evento NewTick não será colocado na fila do programa MQL5.

O evento NewTick é gerado independentemente de se a negociação automática está habilitada ou desabilitada (o botão "Negociação automática"). Ao desabilitar a negociação automática, será apenas desativado o envio de solicitações de negociação a partir do EA, ao mesmo tempo, o EA continua trabalhando.

Ao desativar a negociação automática clicando no botão "Negociação automática", não se interrompe a execução atual da função OnTick().

Exemplo de EA em que toda a lógica de negociação é colocada na função OnTick()

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de EA que opera segundo a direção da vela \"detonante\"
#property description "O corpo da vela \"detonante\" tem um tamanho maior que k*ATR"
#property description "O parâmetro \"revers\" inverte a direção do sinal"

input double lots=0.1;         // volume em lotes
input double kATR=3;           // tamanho da vela de sinal no ATR
input int    ATRperiod=20;     // período do indicador ATR
input int    holdbars=8;       // número de barras para manter a posição
input int    slippage=10;      // derrapagem admissível
input bool   revers=false;     // invertemos o sinal?
input ulong  EXPERT_MAGIC=0;   // MagicNumber do EA
```

```
//--- para armazenar o identificador do indicador ATR
int atr_handle;
//--- aqui vamos armazenar os últimos valores do ATR e o corpo da vela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inicializamos as variáveis globais
   last_atr=0;
   last_body=0;
//--- definimos o volume correto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- criamos o identificador do indicador ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: não foi possível criar o iATR, código de erro %d",__FUNCTION__,
      return(INIT_FAILED);
     }
//--- inicialização bem-sucedida do EA
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- relatamos o código de desligamento do EA
   Print(__FILE__,": Código de motivo da desinicialização = ",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- sinal de negociação
   static int signal=0; // +1 significa um sinal de compra, -1 significa um sinal de v
//--- verificamos e fechamos as posições abertas antigas, abertas há mais de holdbars
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- verificamos o surgimento de uma nova barra
   if(isNewBar())
     {
      //--- verificamos a presença de sinal
      signal=CheckSignal();
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- se aberta uma posição de 'netting', ignoramos o sinal e esperamos até que ele fe
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // saímos do manipulador de eventos NewTick e não entramos no mercado at
     }
//--- para contas de cobertura ('hedge'), casa posição tem vida e fecha separadamente
   if(signal!=0)
     {
      //--- sinal de compra
      if(signal>0)
        {
         PrintFormat("%s: Existe um sinal de compra! Revers=%s",__FUNCTION__,string(re
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- sinal de venda
      if(signal<0)
        {
         PrintFormat("%s: Existe um sinal de venda! Revers=%s",__FUNCTION__,string(rev
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- fim da função OnTick
  }
//+------------------------------------------------------------------+
//| Verificando se á sinal de negociação                             |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 significa que não há sinal
   int res=0;
//--- obtemos o valor do ATR na penúltima barra concluída (o índice da barra igual a 2
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- recebemos os dados da última barra fechada numa matriz do tipo MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- calculamos o tamanho do corpo da barra na última barra fechada
         last_body=bar[0].close-bar[0].open;
         //--- se o corpo da última barra (com índice 1) exceder o valor anterior do A
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // para a leva altista um valor positivo
        }
```

```
      else
         PrintFormat("%s: Não foi possível obter a última barra! Erro",__FUNCTION__,Ge
     }
   else
      PrintFormat("%s: Não foi possível obter o valor do indicador ATR! Erro",__FUNCTI
//--- se estiver ativado o modo de negociação de reversão
   res=revers?-res:res;  // se necessário, revertemos o sinal (em vez de 1, retornamos
//--- retornamos o valor do sinal de negociação
   return (res);
  }
//+------------------------------------------------------------------+
//|  Retornando true quando aparece uma nova barra                   |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // armazenamos o tempo de abertura da barra atual
//--- obtemos o tempo de abertura da barra zero
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- se o tempo de abertura mudar, é porque apareceu uma nova barra
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- exibir no log informações sobre o tempo de abertura da nova barra
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- exibimos uma mensagem sobre o tempo de abertura da nova barra
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- obtemos os dados do último tick
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- exibimos o tempo do último tick em segundos
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- temos uma nova barra
      return (true);
     }
//--- não há nenhuma barra nova
   return (false);
  }
//+------------------------------------------------------------------+
//| Comprando a mercado com o volume especificado                    |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

```
//--- compramos a mercado
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Vendendo a mercado com o volume definido                         |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendemos a mercado
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Fechando posições segundo o tempo de retenção nas barras         |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // número de posições abertas
//--- pesquisa detalhada de todas as posições abertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parâmetros da posição
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- se a posição tem vivido por um longo tempo, e também o MagicNumber e o sím
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // reduzimos o registro do texto para uma correta fo
         PrintFormat("Fechamos a posição #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- definindo o tipo de ordem e de envio do pedido de negociação
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparando e enviando uma solicitação de negociação              |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- declaração e inicialização de estruturas
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- parâmetros da solicitação
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operação de neg
   request.position =pos_ticket;                            // boleta da posição, se i
   request.symbol   =Symbol();                              // símbolo
   request.volume   =volume;                                // volume
   request.type     =type;                                  // tipo de ordem
   request.price    =price;                                 // preço de transação
   request.deviation=slip;                                  // desvio permitido em rel
   request.magic    =magicnumber;                           // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
     {
      //--- exibimos as informações sobre a falha
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- relatamos sobre a operação bem-sucedida
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Veja também

Funções de processamento de eventos, Execução de programas, Eventos do terminal do cliente, OnTimer, OnBookEvent, OnChartEvent

© 2000-2024, MetaQuotes Ltd.

OnCalculate

É chamada em indicadores quando ocorre o evento Calculate para processar alterações nos dados de preço. Existem duas versões da função, você não pode usar as duas dentro do mesmo indicador.

Cálculo baseado numa matriz de dados

```
int  OnCalculate(
   const int        rates_total,       // tamanho da matriz price[]
   const int        prev_calculated,   // número de barras processadas na chamada ante
   const int        begin,             // número do índice na matriz price[] a partir
   const double&    price[]            // matriz de valores para cálculo
   );
```

Cálculos baseados nas séries temporais do timeframe atual

```
int  OnCalculate(
   const int        rates_total,       // tamanho das séries temporais de entrada
   const int        prev_calculated,   // número de barras processadas na chamada ante
   const datetime&  time[],            // matriz Time
   const double&    open[],            // matriz Open
   const double&    high[],            // matriz High
   const double&    low[],             // matriz Low
   const double&    close[],           // matriz Close
   const long&      tick_volume[],     // matriz Tick Volume
   const long&      volume[],          // matriz Real Volume
   const int&       spread[]           // matriz Spread
   );
```

Parâmetros

rates_total

[in] Tamanho da matriz price[] ou das séries temporais de entrada disponíveis para o indicador para cálculo. Na segunda versão da função, o valor do parâmetro corresponde ao número de barras no gráfico em que está sendo executado.

prev_calculated

[in] Contém o valor retornado pela função OnCalculate() na chamada anterior. Ele é usado para ignorar nos cálculos as barras que não mudaram desde a última inicialização desta função.

begin

[in] valor do índice na matriz price[] a partir do qual começam os dados significativos. Permite ignorar nos cálculos os dados ausentes ou iniciais, para os quais não há valores corretos.

price[]

[in] Matriz de valores para executar cálculos. Como matriz price[] pode ser transferida uma das séries temporais de preços ou o buffer calculado de qualquer indicador. O tipo de dados que foi transferido para o cálculo pode ser determinado usando a variável predefinida _AppliedTo.

time{}

[in] Matriz com os valores de tempo da abertura das barras.

© 2000-2024, MetaQuotes Ltd.

open[]

[in] Matriz com os valores dos preços de abertura.

high[]

[in] Matriz com os valores dos preços máximos.

low[]

[in] Matriz com os valores dos preços mínimos.

close[]

[in] Matriz com os valores dos preços de fechamento.

tick_volume[]

[in] Matriz com valores dos volumes de ticks.

volume[]

[in] Matriz com valores dos volumes de negociação.

spread[]

[in] Matriz com os valores de spread para barras.

Valor retornado

Valor de tipo int que será transferido como parâmetro *prev_calculated*, da próxima vez que a função for chamada.

Observação

Se a função OnCalculate() retornar um valor zero, a janela DataWindow do terminal do cliente não exibirá os valores do indicador.

Se os dados de preço foram alterados desde a última chamada da função OnCalculate() (um histórico mais profundo foi carregado ou o histórico estava ausente), o valor do parâmetro de entrada *prev_calculated* será definido como zero pelo próprio terminal.

Para determinar a direção da indexação nas matrizes *time[]*, *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* e *spread[]*, é necessário chamar a função ArrayGetAsSeries(). Para não depender dos padrões, é absolutamente necessário chamar a função ArraySetAsSeries() para aquelas matrizes com as quais se deve trabalhar.

Ao usar a primeira versão da função, a séries temporais ou o indicador necessário como uma matriz price[] é selecionado pelo usuário na guia Parameters quando o indicador é iniciado. Para fazer isso, você deve especificar o elemento desejado na lista suspensa do campo "Apply to".

Para obter os valores do indicador personalizado a partir de outros programas MQL5, é usada a função iCustom() que retorna o identificador do indicador para as operações subsequentes. Neste caso, você pode especificar a matriz necessária *price[]* ou o identificador da outro indicador. Este parâmetro deve ser o último a ser passado na lista de variáveis de entrada do indicador personalizado.

É necessário usar o relacionamento entre o valor retornado pela função OnCalculate() e o segundo parâmetro de entrada *prev_calculated*. O parâmetro *prev_calculated*, ao chamar a função, contém o valor retornado pela função OnCalculate() na chamada anterior. Isso possibilita a implementação de algoritmos econômicos para o cálculo do indicador personalizado, a fim de evitar cálculos repetitivos para as barras que não foram alteradas desde a última inicialização desta função.

© 2000-2024, MetaQuotes Ltd.

Exemplo de indicador

```
//+------------------------------------------------------------------+
//|                                          OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de cálculo do indicador Momentum"

//---- indicator settings
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- parâmetros de entrada
input int MomentumPeriod=14; // Período para cálculo
//---- buffer de indicador
double    MomentumBuffer[];
//--- variável global para armazenar o período de cálculos
int       IntPeriod;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- verificamos o parâmetro de entrada
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("parâmetro Período possui um valor inválido. Para cálculos, será usado o v
     }
   else
      IntPeriod=MomentumPeriod;
//---- buffers
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- o nome do indicador a ser exibido na DataWindow e na subjanela
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- definimos o número da barra a partir da qual começa a plotagem
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- definimos 0.0 como um valor vazio que não é plotado
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- precisão para mostrar os valores do indicador
   IndicatorSetInteger(INDICATOR_DIGITS,2);
```

```
  }
//+------------------------------------------------------------------+
//|  Cálculo do indicador Momentum                                   |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // tamanho da matriz price[]
                const int prev_calculated, // número de barras processadas anteriormen
                const int begin,           // começo dos valores significativos
                const double &price[])     // matriz para o valor a ser processado
  {
//--- posição inicial para cálculos
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- se não houver suficientes dados para cálculo
   if(rates_total<StartCalcPosition)
      return(0);  // saímos com valor nulo - indicador não é calculado
//--- correct draw begin
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- começamos os cálculos, calculamos a posição do início
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- ciclo principal de cálculos
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- execução da OnCalculate concluída. Retornamos o novo valor de prev_calculated pa
   return(rates_total);
  }
```

Veja também

ArrayGetAsSeries, ArraySetAsSeries, iCustom, Funções de processamento de eventos, Execução de programas, Eventos do terminal do cliente, Acesso a séries temporais e indicadores

© 2000-2024, MetaQuotes Ltd.

OnTimer

É chamada em EAs quando o ocorre o evento Timer gerado pelo terminal com um intervalo predeterminado.

```
void  OnTimer(void);
```

Valor retornado

Sem valor retornado

Observação

O evento Timer é gerado periodicamente pelo terminal do cliente para o EA que ativa o temporizador usando a função EventSetTimer(). Geralmente, esta função é chamada na função OnInit(). Ao concluir o trabalho do EA, é necessário destruir o temporizador criado com a ajuda da EventKillTimer() que, normalmente é chamada na função OnDeinit().

Cada EA e cada indicador trabalha com seu temporizador e recebe eventos apenas dele. No final do trabalho do programa MQL5, o temporizador é forçosamente destruído, se foi criado, mas não foi desabilitado pela função EventKillTimer().

Se precisar receber eventos do temporizador mais de uma vez por segundo, use a EventSetMillisecondTimer() para criar um temporizador de alta resolução.

Em geral, à medida que o período do temporizador diminui, o tempo de teste aumenta, uma vez que o número de chamadas do manipulador de eventos do tempo total aumenta. Ao trabalhar em tempo real, os eventos do temporizador são gerados no máximo 1 vez a cada 10-16 milissegundos, o que é devido a limitações de hardware.

Para cada programa, não pode ser iniciado mais do que um temporizador. Cada programa MQL5 e cada gráfico têm sua própria fila de eventos, à qual são adicionados todos os eventos recém-chegados. Se na fila já houver um evento Timer ou este evento estiver no estado de processamento, o novo evento Timer não será colocado na fila do programa MQL5.

Exemplo de EA com manipulador OnTimer()

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de uso de temporizador para calcular a hora do servido
#property description "É melhor iniciar o EA no final da semana de negociação, antes d
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- criamos um temporizador com um período de 1 segundo
```

```
   EventSetTimer(1);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destruímos o temporizador no final do trabalho
   EventKillTimer();

  }
//+----------´-------------------------------------------------------+
//| Expert tck function                                              |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- hora da primeira chamada da OnTimer()
   static datetime start_time=TimeCurrent();
//--- hora do servidor de negociação na primeira chamada da OnTimer();
   static datetime start_tradeserver_time=0;
//--- hora do servidor de negociação calculada
   static datetime calculated_server_time=0;
//--- hora local no computador
   datetime local_time=TimeLocal();
//--- hora estimada atual do servidor de negociação
   datetime trade_server_time=TimeTradeServer();
//--- se por algum motivo a hora do servidor for desconhecida, sairemos antecipadament
   if(trade_server_time==0)
      return;
//--- se o valor inicial do servidor de negociação ainda não estiver definido
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- definimos a hora calculada do servidor de negociação
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- aumentamos o tempo da primeira chamada da OnTimer()
       if(start_tradeserver_time!=0)
          calculated_server_time=calculated_server_time+1;;
      }
//---
   string com=StringFormat("                    Start time: %s\r\n",TimeToString(start_t
   com=com+StringFormat("                   Local time: %s\r\n",TimeToString(local_time
   com=com+StringFormat("TimeTradeServer time: %s\r\n",TimeToString(trade_server_time,
   com=com+StringFormat(" EstimatedServer time: %s\r\n",TimeToString(calculated_server
//--- exibimos no gráfico os valores de todos os contadores
   Comment(com);
  }
```

Veja também

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Eventos do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnTrade

É chamada em EAs quando ocorre o evento Trade. A função se destina ao processamento de alterações em listas de ordens, de posições e de transações.

```
void  OnTrade(void);
```

Valor retornado

Sem valor retornado

Observação

A OnTrade() é chamada apenas para EAs, é ignorada em indicadores e em scripts, mesmo se você adicionar uma função com esse nome e tipo a eles.

Para qualquer ação de negociação (colocação de uma ordem pendente, abertura/fechamento de uma posição, definição de stops, desencadeamento de ordens pendentes, etc.), são devidamente modificados o histórico de ordens e de transações e/ou a lista de posições e de ordens atuais.

Ao processar a ordem, o servidor de negociação envia uma mensagem ao terminal sobre a ocorrência do evento Trade. Para obter a partir do histórico dados atuais sobre ordens e transações, você deve primeiro executar uma solicitação de histórico de negociação usando HistorySelect().

Eventos de negociação são gerados pelo servidor nos seguintes casos:

- alterações nas ordens atuais,
- alterações nas posições,
- alterações nas transações,
- alterações no histórico de negociação.

Cada evento Trade pode ser o resultado de uma ou mais solicitações de negociação. As solicitações de negociação são enviadas para o servidor usando OrderSend() ou OrderSendAsync(). Cada solicitação pode gerar vários eventos de negociação. Você não pode confiar na regra «Uma solicitação - Um evento 'Trade'», já que o processamento de solicitações pode ocorrer em vários estágios e cada operação pode alterar o status de ordens, de posições e do histórico de negociação.

O manipulador OnTrade() é chamado após a chamadas correspondentes da OnTradeTransaction(). Em geral, não há correlação exata no número de chamadas da OnTrade() e da OnTradeTransaction(). Uma chamada da OnTrade() corresponde a uma ou mais chamadas da OnTradeTransaction.

Exemplo de EA com manipulador OnTrade()

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
input    int days=7;            // profundidade do histórico de negociação em dias
//--- definimos no nível global os limites do histórico de negociação
datetime     start;             // data de início do histórico de negociação em cache
datetime     end;               // data final do histórico de negociação em cache
//--- contadores globais
int          orders;            // número de ordens vigentes
int          positions;         // número de posições abertas
int          deals;             // número de transações no cache do histórico de negoc
int          history_orders;    // número de ordens no cache do histórico de negociaçã
bool         started=false;     // sinalizador da relevância dos contadores

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Limites do histórico de negociação carregado: início - %s, fim - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  Inicializando contadores de posições, de ordens e de transações |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- carregamos o histórico
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Não foi possível carregar no cache o histórico de %s a %s. Cód
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- obtemos os valores atuais
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("Inicialização de contadores de ordens, de posições e de transações bem-suce
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| chamado quando ocorre o evento Trade                             |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| exemplo de processamento de alterações na negociação e no histórico
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- carregamos no cache do programa o histórico de negociação a partir do intervalo
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Não foi possível carregar no cache o histórico de %s a %s. Cód:
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- obtemos os valores atuais
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- verificamos as alterações na quantidade de ordens vigentes
   if(curr_orders!=orders)
     {
      //--- número de ordens vigentes alterado
      PrintFormat("O número de ordens foi alterado de %d para %d",
                  orders,curr_orders);
      //--- atualizamo o valor
      orders=curr_orders;
     }
//--- alteração no número de posições abertas
   if(curr_positions!=positions)
     {
      //--- o número de posições abertas foi alterado
      PrintFormat("O número de posições abertas foi alterado de %d para %d",
```

© 2000-2024, MetaQuotes Ltd.

```
                    positions,curr_positions);
      //--- atualizamo o valor
      positions=curr_positions;
     }
//--- alterações no número de transações no cache do histórico de negociação
   if(curr_deals!=deals)
     {
      //--- número de transações no cache do histórico de negociação foi alterado
      PrintFormat("O número de transações foi alterado de %d para %d",
                  deals,curr_deals);
      //--- atualizamo o valor
      deals=curr_deals;
     }
//--- alterações no número de ordens históricas no cache do histórico de negociação
   if(curr_history_orders!=history_orders)
     {
      //--- número de ordens históricas no cache do histórico de negociação foi altera
      PrintFormat("O número de ordens no histórico foi alterado de %d para %d",
                  history_orders,curr_history_orders);
     //--- atualizamos o valor
     history_orders=curr_history_orders;
     }
//--- verificamos se é necessário alterar os limites do histórico de negociação para s
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  alterações da data de início para a solicitação do histórico de negociação
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- intervalo de início, se começarmos a trabalhar agora
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- verificamos que o limite do início do histórico de transações seja inferior
//--- a 1 dia a partir da data planejada
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- deve-se corrigir a data de início do histórico carregado no cache
      start=curr_start;
      PrintFormat("Novo limite de início do histórico de negociação carregado: início
                  TimeToString(start));
      //--- agora recarregamos o histórico de transações para o intervalo atualizado
      HistorySelect(start,end);
      //--- corrigimos os contadores de transações e de ordens no histórico para a pro
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* Exemplo de exibição:
```

```
  Limites do histórico de negociação carregado: início - 2018.07.16 18:11, fim - 2018.
  Contadores de ordens, deposições e de transações inicializados com sucesso
  O número de ordens foi alterado. Havia 0, há 1
  O número de ordens foi alterado. Havia 1, há 0
  O número de posições foi alterado. Havia 0, há 1
  O número de transações foi alterado. Havia 0, há 1
  O número de ordens no histórico foi alterado. Havia 0, há 1
*/
```

Veja também

OrderSend, OrderSendAsync, OnTradeTransaction, Eventos do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnTradeTransaction

É chamada em EAs quando ocorre o evento TradeTransaction. A função é projetada para processar os resultados de execução da solicitação de negociação.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // estrutura da transação comercial
   const MqlTradeRequest&        request,   // estrutura da solicitação
   const MqlTradeResult&         result     // estrutura da resposta
   );
```

Parâmetros

trans

[in] Variável do tipo MqlTradeTransaction com descrição da transação feita na conta de negociação.

request

[in] Variável do tipo MqlTradeRequest com descrição da solicitação que gerou a transação. Contém os valores apenas para a transação de tipo TRADE_TRANSACTION_REQUEST.

result

[in] Variável do tipo MqlTradeResult com resultado da execução da solicitação que gerou a transação. Contém os valores apenas para a transação de tipo TRADE_TRANSACTION_REQUEST.

Valor retornado

Sem valor retornado

Observação

OnTradeTransaction() é chamada para manipular o evento TradeTransaction que o servidor de negociação envia para o terminal nos seguintes casos:

- envio da solicitação de negociação a partir do programa MQL5 usando as funções OrderSend()/OrderSendAsync() e sua posterior execução;
- envio manual da solicitação de negociação através da interface gráfica e sua posterior execução;
- desencadeamento de ordens pendentes e ordens stop no servidor;
- execução de operações no lado do servidor de negociação.

Informações sobre o tipo de transação são contidas no campo *type* da variável *trans*. Os tipos de trasações comerciais são descritas na enumeração ENUM_TRADE_TRANSACTION_TYPE:

- TRADE_TRANSACTION_ORDER_ADD - adição de uma nova ordem ativa
- TRADE_TRANSACTION_ORDER_UPDATE - alteração da ordem ativa
- TRADE_TRANSACTION_ORDER_DELETE - remoção da ordem da lista de ordens ativas
- TRADE_TRANSACTION_DEAL_ADD - adição da transação ao histórico
- TRADE_TRANSACTION_DEAL_UPDATE - alteração da transação no histórico
- TRADE_TRANSACTION_DEAL_DELETE - remoção da transação do histórico
- TRADE_TRANSACTION_HISTORY_ADD - adição da ordem ao histórico como resultado da execução ou do cancelamento
- TRADE_TRANSACTION_HISTORY_UPDATE - alteração da ordem que se encontra no histórico de ordens

© 2000-2024, MetaQuotes Ltd.

- TRADE_TRANSACTION_HISTORY_DELETE – remoção da ordem do histórico de ordens
- TRADE_TRANSACTION_POSITION – alteração da posição não relacionada à execução da transação
- TRADE_TRANSACTION_REQUEST – notificação de que a solicitação de negociação foi processada pelo servidor e o resultado de seu processamento foi recebido.

Ao processar transações do tipo TRADE_TRANSACTION_REQUEST, para obter informações adicionais, é necessário analisar os segundo e terceiro parâmetros da função OnTradeTransaction() – *request* e *result*.

O envio de uma solicitação de compra leva a uma cadeia de transações comerciais feitas na conta de negociação: 1) a solicitação é aceita para ser processada, 2) para a conta é criada a respectiva solicitação de compra, 3) é executada a ordem, 4) a ordem executada é removida da lista de ordens ativas, 5) adicionada ao histórico de ordens, 6) a transação correspondente é adicionada ao histórico e 7) é criada uma nova posição. Todas essas ações são transações comerciais. A chegada de cada transação desse tipo no terminal é um evento TradeTransaction. FFNeste caso, não é garantida a ordem de recepção destas transações ao terminal, portanto, não se pode construir seu próprio algoritmo de negociação na expectativa de receber certas transações comerciais após a chegada de outras.

Quando EA processa transações comerciais usando o manipulador OnTradeTransaction(), o terminal continua processando as transações recém-chegadas. Assim, o estado da conta de negociação pode mudar já durante o funcionamento da OnTradeTransaction(). Por exemplo, enquanto o programa MQL5 manipula o evento de adição de nova ordem, ela pode ser executada, excluída da lista de pedidos ordens abertas e movida para o histórico. No futuro, o programa será notificado de todos esses eventos.

O comprimento da fila de transações é de 1 024 itens. Se a OnTradeTransaction() demorar a processar a seguinte transação muito tempo, as transações anteriores na fila poderão ser substituídas por transações mais recentes.

O manipulador OnTrade() é chamado após as respectivas chamadas da OnTradeTransaction(). Em geral, não há correlação exata no número de chamadas da OnTrade() e da OnTradeTransaction(). Uma chamada da OnTrade() corresponde a uma ou mais chamadas da OnTradeTransaction.

Cada evento Trade pode ser o resultado de uma ou mais solicitações de negociação. As solicitações de negociação são enviadas para o servidor usando OrderSend() ou OrderSendAsync(). Cada solicitação pode gerar vários eventos de negociação. Você não pode confiar na regra «Uma solicitação - Um evento 'Trade'», já que o processamento de solicitações pode ocorrer em vários estágios e cada operação pode alterar o status de ordens, de posições e do histórico de negociação.

Exemplo de EA com manipulador OnTradeTransaction()

```
//+------------------------------------------------------------------+
//|                                   OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de ouvinte do evento TradeTransaction"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   PrintFormat("LAST PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // contador de chamadas da OnTradeTransaction()
   static uint lasttime=0; // hora da última chamada da OnTradeTransaction()
//---
   uint time=GetTickCount();
//--- se a última operação tiver sido realizada há mais de 1 segundo,
   if(time-lasttime>1000)
     {
      counter=0; // significa que se trata de uma nova operação de negociação e, porta
      if(IS_DEBUG_MODE)
         Print(" Nova operação de negociação");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- resultado da execução do pedido de negociação
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- nome do símbolo segundo o qual foi realizada a transação
   string trans_symbol=trans.symbol;
//--- tipo de transação
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
```

```
      case  TRADE_TRANSACTION_POSITION:   // alteração da posição
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%d %s modified: SL=%.5f TP=%.5f"
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
      case TRADE_TRANSACTION_REQUEST:     // envio do pedido de negociação
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:    // adição da transação
        {
         ulong           lastDealID   =trans.deal;
         ENUM_DEAL_TYPE lastDealType =trans.deal_type;
         double         lastDealVolume=trans.volume;
         //--- identificador da transação no sistema externo - bilhete atribuído pela
         string Exchange_ticket="";
         if(HistoryDealSelect(lastDealID))
            Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

         PrintFormat("MqlTradeTransaction: %s deal #%d %s %s %.2f lot   %s",EnumToStri
                     lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
        }
      break;
      case TRADE_TRANSACTION_HISTORY_ADD: // adição da ordem ao histórico
        {
         //--- identificador da transação no sistema externo - bilhete atribuído pela
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_FILLED)
           {
            if(HistoryOrderSelect(lastOrderID))
               Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("(Exchange ticket=%s)",Exchange_ticket);
           }
         PrintFormat("MqlTradeTransaction: %s order #%d %s %s %s   %s",EnumToString(ti
                     lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToString
        }
      break;
      default: // outras transações
        {
         //--- identificador da ordem no sistema externo - bilhete atribuído pela Bols
         string Exchange_ticket="";
         if(lastOrderState==ORDER_STATE_PLACED)
           {
            if(OrderSelect(lastOrderID))
               Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
```

© 2000-2024, MetaQuotes Ltd.

```
            if(Exchange_ticket!="")
               Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
            }
         PrintFormat("MqlTradeTransaction: %s order #%d %s %s   %s",EnumToString(trans
                     lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
         }
      break;
      }
//--- bilhete da ordem
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: order #%d retcode=%s ",orderID_result,retcode_resul
//---
  }
//+------------------------------------------------------------------+
//| converte códigos numéricos de respostas em códigos Mnemonic de string
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
```

© 2000-2024, MetaQuotes Ltd.

```
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");        break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");        break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");       break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");     break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

Veja também

OrderSend, OrderSendAsync, OnTradeTransaction, Estrutura do pedido de negociação, Estrutura da transação comercial, Tipos de transações comerciais, Tipos de operações de negociação, Eventos do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnBookEvent

É chamada em indicadores e EAs quando ocorre o evento BookEvent. A função é projetada para processar alterações no livro de ofertas (Depth of Market).

```
void  OnBookEvent(
   const string&  symbol         // símbolo
   );
```

Parâmetros

symbol

[in] Nome do instrumento financeiro em que ocorre o evento BookEvent

Valor retornado

Sem valor retornado

Observação

Para receber eventos BookEvent em qualquer símbolo, basta fazer uma pré-assinatura a fim de receber esses eventos para este símbolo usando a função MarketBookAdd(). Para cancelar a assinatura de recebimento do evento BookEvent num símbolo específico, você deve chamar a função MarketBookRelease().

O evento BookEvent é difundido no gráfico. Isso significa que basta que um aplicativo no gráfico assine a recepção do evento BookEvent usando a função MarketBookAdd, para todos os outros indicadores e EAs - que estão sendo executados neste gráfico e que têm o manipulador OnBookEvent() - receberem esse evento. Por isso, você precisa analisar o nome do símbolo que é transferido para o manipulador OnBookEvent() como um parâmetro *symbol*.

Para todos os aplicativos em andamento no mesmo gráfico, são executados contadores separados para receber eventos BookEvent na seção de símbolos. Isso significa que em cada gráfico pode haver assinaturas de diferentes símbolos, e para cada símbolo existe seu próprio contador. Assinatura e remoção de assinatura de eventos BookEvent alteram o contador de assinaturas só dos símbolos especificados, apenas dentro do mesmo gráfico. Isso significa que, em dois gráficos adjacentes, pode haver assinaturas de eventos BookEvent para o mesmo símbolo, mas com valores diferentes para o contador de assinaturas.

O valor inicial do contador de assinaturas é zero. Após cada chamada da MarketBookAdd(), o contador de assinaturas para o símbolo especificado neste gráfico é aumentado em 1 (não é necessário o símbolo do gráfico e o símbolo na MarketBookAdd() coincidirem). Após a chamada da MarketBookRelease(), o contador de assinaturas para o símbolo especificado dentro do gráfico é reduzido em 1. A transmissão de eventos BookEvent em qualquer símbolo dentro do gráfico continua até que o contador de assinaturas para este símbolo se torne zero. Portanto, é importante que cada programa MQL5 contendo chamadas da MarketBookAdd(), ao concluir seu trabalho, corretamente cancele a recepção de eventos para cada símbolo com a ajuda da MarketBookRelease(). Para isso, basta que - para cada chamada - o número de chamadas da MarketBookAdd() e da MarketBookRelease() seja par durante toda a vida útil do programa MQL5. Usar sinalizadores ou próprios contadores de assinaturas dentro do programa permite trabalhar com segurança com eventos BookEvent e impede a remoção de assinaturas de recepção desse evento em programas de terceiros dentro do mesmo gráfico.

© 2000-2024, MetaQuotes Ltd.

Os eventos BookEvent nunca são ignorados e sempre são enfileirados, mesmo que o processamento do evento BookEvent anterior ainda não tenha sido concluído. Deve-se ter em mente que os eventos BookEvent são entregues por eles mesmos e não levam consigo o status do livro de ofertas. Isso significa que a chamada da MarketBookGet() a partir do manipulador OnBookEvent() permite - em vez do status do livro de ofertas que causou o envio do evento BookEvent - obter o status atual do livro de ofertas no momento da chamada. Para garantir que todos os status exclusivos sejam obtidos, a função OnBookEvent() deve ser o mais rápida possível.

Exemplo

```
//+------------------------------------------------------------------+
//|                                          OnBookEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/pt/articles/2635"
#property version   "1.00"
#property description "Exemplo de medição da velocidade de atualização do livro de ofe
#property description "Código é retirado do artigo https://www.mql5.com/pt/articles/26
//--- parâmetros de entrada
input ulong ExtCollectTime   =30;  // tempo de teste em segundos
input ulong ExtSkipFirstTicks=10;  // número de ticks ignorados no início
//--- sinalizador da presença de assinatura de recepção de eventos BookEvent
bool book_subscribed=false;
//--- matriz para receber solicitações a partir do livro de ofertas
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- definimos o início
   Comment(StringFormat("Aguardamos a recepção dos primeiros %I64u ticks",ExtSkipFirst
   PrintFormat("Aguardamos a recepção dos primeiros %I64u ticks",ExtSkipFirstTicks);
//--- ativamos a transmissão do livro de ofertas
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: Função MarketBookAdd(%s) retornou true",__FUNCTION__,_Symbol);
     }
   else
      PrintFormat("%s: Função MarketBookAdd(%s) retornou false! GetLastError()=%d",__F
//--- inicialização bem-sucedida
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialize expert                                              |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- exibimos o código de motivo da desinicialização
   Print(__FUNCTION__,": Código de motivo da desinicialização = ",reason);
//--- cancelamos nossa assinatura de recepção de eventos a partir do livro de ofertas
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s) retornou false! GetLastError()=%d",_Sy
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent function                                               |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;             // tempo de início do teste
   static ulong tickcounter=0;           // contador de atualização do livro de oferta
//--- trabalhamos com eventos do livro de ofertas somente se ele tiver sido assinado p
   if(!book_subscribed)
      return;
//--- consideramos atualizações apenas do próprio símbolo
   if(symbol!=_Symbol)
      return;
//--- pulamos os primeiros ticks para a limpeza primária da fila e para aquecimento
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- lembramo-nos do tempo de início
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- consultamos os dados do livro de ofertas
   MarketBookGet(symbol,book);
//--- quando se deve parar?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// quanto tempo passou em microssegundos desde o início do teste?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endtime/1000.0
      ExpertRemove();
      return;
     }
//--- exibição dos contadores no campo de comentários
```

© 2000-2024, MetaQuotes Ltd.

```
   if(endtime>0)
      Comment(StringFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endt
   }
```

Veja também

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, Função de manipulação de eventos, Execução do programa, Eventos do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnChartEvent

É chamada em EAs e indicadores quando ocorre o evento ChartEvent. A função é projetada para processar mudanças no gráfico causadas por ações do usuário ou pelo funcionamento de programas MQL5

```
void  OnChartEvent()
   const int       id,       // identificador de evento
   const long&     lparam,   // parâmetro do evento de tipo long
   const double&   dparam,   // parâmetro do evento de tipo double
   const string&   sparam    // parâmetro do evento de tipo string
   );
```

Parâmetros

id

[in] Identificador do evento a partir da enumeração ENUM_CHART_EVENT.

lparam

[in] Parâmetro do evento de tipo long

dparam

[in] Parâmetro do evento de tipo double

sparam

[in] Parâmetro do evento de tipo string

Valor retornado

Sem valor retornado

Observação

Existem 11 tipos de eventos que podem ser processados usando a função predefinida OnChartEvent() . Para eventos personalizados, são fornecidos 65 535 identificadores no intervalo de CHARTEVENT_CUSTOM a CHARTEVENT_CUSTOM_LAST. Para gerar um evento personalizado, você deve usar a função EventChartCustom().

Breve descrição dos eventos da enumeração ENUM_CHART_EVENT:

- CHARTEVENT_KEYDOWN — pressionamento do teclado quando a janela do gráfico está em foco;
- CHARTEVENT_MOUSE_MOVE — deslocamento do mouse e pressionamento dos botões do mouse (se para o gráfico estiver definida a propriedade CHART_EVENT_MOUSE_MOVE=true);
- CHARTEVENT_OBJECT_CREATE — criação do objeto gráfico (se para o gráfico estiver definida a propriedade CHART_EVENT_OBJECT_CREATE=true);
- CHARTEVENT_OBJECT_CHANGE — alteração das propriedades do objeto através do diálogo de propriedades;
- CHARTEVENT_OBJECT_DELETE — remoção do objeto gráfico (se para o gráfico estiver definida a propriedade CHART_EVENT_OBJECT_DELETE=true);
- CHARTEVENT_CLICK — clique do mouse no gráfico;
- CHARTEVENT_OBJECT_CLICK — clique do mouse no objeto gráfico que pertence ao gráfico;
- CHARTEVENT_OBJECT_DRAG — deslocamento do objeto gráfico com o mouse;

© 2000-2024, MetaQuotes Ltd.

- CHARTEVENT_OBJECT_ENDEDIT — fim da edição de texto no campo inserido do objeto gráfico Edit (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — alteração do gráfico;
- CHARTEVENT_CUSTOM+n — identificador do evento personalizado, onde n está no intervalo de 0 a 65 535. CHARTEVENT_CUSTOM_LAST contém o último identificador válido de evento personalizado (CHARTEVENT_CUSTOM+65535).

Todos os programas MQL5 trabalham em fluxos diferentes do fluxo principal do aplicativo. O fluxo principal do terminal é responsável pelo processamento de todas as mensagens de sistema do Windows e, como resultado dele, gera mensagens do Windows para seu próprio aplicativo. Por exemplo, o deslocamento do mouse no gráfico (evento WM_MOUSE_MOVE) gera várias mensagens de sistema para a plotagem subsequente da janela do aplicativo, bem como envia mensagens internas para EAs e para indicadores que estão sendo executados neste gráfico. Neste caso, pode acontecer que o fluxo principal do aplicativo ainda não consiga processar a mensagem de sistema WM_PAINT (e, portanto, ainda não plote a gráfico alterado), enquanto o EA ou indicador já tenha recebido o evento sobre o movimento do cursor do mouse. Assim, a propriedade do gráfico CHART_FIRST_VISIBLE_BAR será alterada somente após a plotagem do gráfico.

Para cada tipo de evento, os parâmetros de entrada da função OnChartEvent() possuem determinados valores necessários para manipular este evento. A tabela lista os eventos e valores que são transferidos através dos parâmetros.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de teclado	CHARTEVENT_KEYDOWN	código de tecla pressionada	Número de pressionamentos da tecla gerados durante o tempo em que é mantida pressionada	Valor de string da bitmask que descreve o status dos botões do teclado
Eventos do mouse (se para o gráfico estiver definida a propriedade CHART_EVENT_MOUSE_MOVE=true)	CHARTEVENT_MOUSE_MOVE	Coordenada X	Coordenada Y	Valor de string da bitmask que descreve o status dos botões do mouse
Evento da roda do mouse (se para o gráfico estiver definida a propriedade CHART_EVENT_MOUSE_WHEEL=true)	CHARTEVENT_MOUSE_WHEEL	Sinalizadores de estados das teclas e dos botões do mouse, coordenadas X e Y do cursor. A descrição é dada no exemplo	Valor Delta de rolagem da roda do mouse	—

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de criação do objeto gráfico (se para o gráfico for definida a propriedade CHART_EVENT_OBJECT_CREATE =true)	CHARTEVENT_OBJECT_CREATE	—	—	Nome do objeto gráfico criado
Evento de alteração das propriedades do objeto através do diálogo de propriedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nome do objeto gráfico alterado
Evento de remoção do objeto gráfico (se para o gráfico estiver definida a propriedade CHART_EVENT_OBJECT_DELETE =true)	CHARTEVENT_OBJECT_DELETE	—	—	Nome do objeto gráfico excluído
Evento de clique no gráfico	CHARTEVENT_CLICK	Coordenada X	Coordenada Y	—
Evento de clique do mouse no objeto gráfico	CHARTEVENT_OBJECT_CLICK	Coordenada X	Coordenada Y	Nome do objeto gráfico no qual ocorreu o evento
Evento de movimento do objeto gráfico com o mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome do objeto gráfico movido
Evento de fim da edição de texto no campo inserido do objeto gráfico "Caixa de edição"	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome do objeto gráfico "Campo de entrada" no qual foi concluída a edição de texto

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de redimensionamento do gráfico ou alteração das propriedades do gráfico através do diálogo de propriedades	CHARTEVENT_CHART_CHANGE	—	—	—
Evento personalizado com número N	CHARTEVENT_CUSTOM+N	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()

Exemplo de um ouvinte dos eventos do gráfico:

```
//+------------------------------------------------------------------+
//|                                          OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo do ouvinte dos eventos do gráfico e do gerador de event
//--- identificadores das teclas Enter, Shift, Caps Lock, Space, Backspace, Ctrl, Alt
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- exibimos o valor da constante CHARTEVENT_CUSTOM
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Executado o EA com nome ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- definindo o sinalizador de recepção dos eventos de criação de objetos do gráfico
```

```
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- definindo o sinalizador de recepção dos eventos de remoção de objetos do gráfic
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- ativação da mensagem de rolagem de roda do mouse
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- a atualização forçada das propriedades do gráfico garante a prontidão para o pr
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- contador de ticks para gerar o evento personalizado
   static int tick_counter=0;
//--- vamos dividir os ticks acumulados por este número
   int simple_number=113;
//---
   tick_counter++;
//--- enviamos o m evento personalizado se o contador de ticks for um múltiplo de sim
   if(tick_counter%simple_number==0)
     {
      //--- formaremos o identificador do evento personalizado no intervalo de 0 a 65
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  enviamos um evento personalizado com preenchimento dos parâmetros
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- fazemos a exibição do log para estudar e analisar os resultados do exempl
      Print(__FUNCTION__,": Evento personalizado enviado com ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- pressionamento de tecla
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("KEY_NUMLOCK_LEFT pressionado");   break;
         case KEY_LEFT:          Print("KEY_LEFT pressionado");           break;
         case KEY_NUMLOCK_UP:    Print("KEY_NUMLOCK_UP pressionado");     break;
```

© 2000-2024, MetaQuotes Ltd.

```
         case KEY_UP:            Print("KEY_UP pressionado");            break;
         case KEY_NUMLOCK_RIGHT: Print("KEY_NUMLOCK_RIGHT pressionado"); break;
         case KEY_RIGHT:         Print("KEY_RIGHT pressionado");         break;
         case KEY_NUMLOCK_DOWN:  Print("KEY_NUMLOCK_DOWN pressionado");  break;
         case KEY_DOWN:          Print("KEY_DOWN pressionado");          break;
         case KEY_NUMPAD_5:      Print("KEY_NUMPAD_5 pressionado");      break;
         case KEY_NUMLOCK_5:     Print("KEY_NUMLOCK_5 pressionado");     break;
         default:                Print("Tecla não listada pressionada");
        }
     }
//--- clique com o botão esquerdo do mouse no gráfico
   if(id==CHARTEVENT_CLICK)
      Print("Coordenadas do clique do mouse no gráfico: x = ",lparam,"  y = ",dparam);
//--- clique no objeto gráfico
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Clique no objeto com nome '"+sparam+"'");
//--- objeto excluído
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Excluído objeto com nome ",sparam);
//--- objeto criado
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("Criado objeto com nome ",sparam);
//--- objeto alterado
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Alterado objeto com nome ",sparam);
//--- objeto deslocado ou coordenadas dos pontos de ancoragem alteradas
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Alteração de pontos de ancoragem do objeto com nome ",sparam);
//--- alterado o texto no campo inserido do objeto gráfico Edit
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Alterado o texto no objeto Edit ",sparam,"  id=",id);
//--- eventos de movimento do mouse
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- analisemos o estado dos botões e da roda do mouse para este evento
      int flg_keys = (int)(lparam>>32);          // sinalizador de estado de teclas Ct
      int x_cursor = (int)(short)lparam;         // coordenada X, em que acontece o ev
      int y_cursor = (int)(short)(lparam>>16);   // coordenada Y, em que acontece o ev
      int delta    = (int)dparam;                // valor total de rolagem da roda, é
      //--- processamos o sinalizador
      string str_keys="";
      if((flg_keys&0x0001)!=0)
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
```

© 2000-2024, MetaQuotes Ltd.

```
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- redimensionando o gráfico ou alterando as propriedades do gráfico através do dia
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("Redimensionamento ou alteração das propriedades do gráfico");
//--- evento personalizado
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Evento personalizado ID=%d, lparam=%d, dparam=%G, sparam=%s",id,lpa
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
```

Veja também

EventChartCustom, Tipos de eventos do gráfico, Funções de processamento de eventos, Execução do programa, Evento do terminal do cliente

© 2000-2024, MetaQuotes Ltd.

OnTester

É chamada em EAs quando ocorre o evento Tester para executar as ações necessárias no final do teste.

```
double  OnTester(void);
```

Valor retornado

Valor de um critério de otimização personalizado para avaliar os resultados do teste.

Observação

A função OnTester() só pode ser usada em EAs durante o teste e é projetada principalmente para calcular um determinado valor a ser usado como critério "Custom max" ao otimizar parâmetros de entrada.

Durante a otimização genética, a classificação de resultados numa geração é feita em ordem decrescente. Isso significa que, do ponto de vista do critério de otimização, os melhores são os resultados com o maior valor. Os piores valores para esta classificação são colocados no final e, posteriormente, descartados, não participando da formação da próxima geração.

Assim, usando a função OnTester(), você pode não apenas criar e salvar seus próprios relatórios de resultados de testes, mas também controlar o processo de otimização para encontrar os melhores parâmetros da estratégia de negociação.

Exemplo de cálculo de um critério de otimização personalizado. A ideia é calcular a regressão linear do gráfico de saldo descrita no artigo Otimizando uma estratégia usando o gráfico do saldo e comparando os resultados com o critério "Balance + max Sharpe Ratio"

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de EA com manipulador OnTester()"
#property description "Como critério de otimização personalizado "
#property description "é retornado o coeficiente de regressão linear do gráfico de sal
#property description "dividido pelo erro quadrático médio do desvio"
//--- conectamos a classe de operações de negociação
#include <Trade\Trade.mqh>
//--- parâmetros de entrada do EA
input double Lots               = 0.1;      // Volume
input int    Slippage           = 10;       // Derrapagem admissível
input int    MovingPeriod       = 80;       // Período de média móvel
input int    MovingShift        = 6;        // Deslocamento de média móvel
//--- variáveis globais
int    IndicatorHandle=0;  // identificador do indicador
bool   IsHedging=false;    //sinal da conta
```

```
CTrade trade;              // para realizar operações de negociação
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Verificando as condições para abertura da posição                |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- negociamos apenas no início da barra nova
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
//--- volume de ticks
   if(rt[1].tick_volume>1)
      return;
//--- obtemos os valores da média móvel
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- verificamos a presença do sinal
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- a vela se abriu mais alto, mas fechou abaixo da média móvel
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // sinal de compra
   else // a vela se abriu mais abaixo, mas fechou acima da média móvel
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// sinal de venda
     }
//--- verificações adicionais
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ? SYMBOL_BID:SY
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Verificando as condições de fechamento da posição                |
//+------------------------------------------------------------------+
```

```
void CheckForClose(void)
  {
   MqlRates rt[2];
//--- negociamos apenas no início da barra nova
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- obtemos os valores da média móvel
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- posição já foi selecionada usando PositionSelect()
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- a vela se abriu mais acima, mas fechou abaixo da média móvel - fechamos a posiçã
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- a vela se abriu mais abaixo, mas fechou acima da média móvel - fechamos a posiçã
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- verificações adicionais
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Selecionamos a posição com base no tipo de conta: Netting ou Hedging
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- seleção da posição para a conta Hedging
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
```

```
            {
             res=true;
             break;
            }
         }
      }
//--- seleção da posição para a conta Netting
   else
      {
       if(!PositionSelect(_Symbol))
          return(false);
       else
          return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---verificação do Mag
      }
//--- resultado da execução
   return(res);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- definimos o tipo de negociação: Netting ou Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- inicializamos o objeto para o controle correto das posições
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- criamos o indicador Moving Average
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
      {
       printf("Erro ao criar o indicador iMA");
       return(INIT_FAILED);
      }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- se a posição já estiver aberta, verificamos condição de fechamento
   if(SelectPosition())
      CheckForClose();
// verificamos a condição para abertura da posição
   CheckForOpen();
```

© 2000-2024, MetaQuotes Ltd.

```
//---
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- valor do critério de otimização personalizado (quanto mais, melhor)
   double ret=0.0;
//--- obtemos os resultados dos trades na matriz
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- se há menos de 10 trades, o teste não gerou resultados positivos
   if(trades<10)
      return (0);
//--- resultado médio no trade
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- exibimos uma mensagem para o modo de teste único
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Trades=%d, Lucro médio=%.2f",__FUNCTION__,trades,average_pl);
//--- calculamos os coeficientes de regressão linear para o gráfico de lucro
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- calculamos o erro de desvio do gráfico em relação à linha de regressão
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- calculamos o rácio do lucro de tendência em relação ao desvio padrão
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- retornamos o valor do critério de otimização personalizado
   return(ret);
  }
//+------------------------------------------------------------------+
//| Obtendo a matriz de lucros/perdas de transações                  |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- consultamos o histórico de negociação completo
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
//--- definimos o tamanho inicial da matriz pelo número de transações no histórico
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayResize(pl_results,total_deals);
//--- contador de trades que fixam o resultado da negociação - lucro ou perda
   int counter=0;
   ulong ticket_history_deal=0;
//--- passar por todos os trades
   for(uint i=0;i<total_deals;i++)
     {
      //--- selecionamos o trade
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket_hi
         long            deal_type   =HistoryDealGetInteger(ticket_history_deal,DEAL_T
         double          deal_profit =HistoryDealGetDouble(ticket_history_deal,DEAL_PR
         double          deal_volume =HistoryDealGetDouble(ticket_history_deal,DEAL_VO
         //--- estamos interessados apenas em operações de negociação
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- somente trades com fixação de lucro/perda
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- escrevemos o resultado da negociação na matriz e aumentamos o contad
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- definimos o tamanho final da matriz
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Calculando a regressão linear de tipo y=a*x+b                    |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- verificamos se há suficientes dados
   if(ArraySize(change)<3)
      return (false);
//--- criamos a matriz do gráfico com acumulação
   int N=ArraySize(change);
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- agora calculamos os coeficientes de regressão
   double x=0,y=0,x2=0,xy=0;
   for(int i=0;i<N;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       x=x+i;
       y=y+chartline[i];
       xy=xy+i*chartline[i];
       x2=x2+i*i;
      }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calcula o erro quadrático médio do desvio para os a e b definidos
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- soma dos quadrados dos erros
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

Veja também

Teste de estratégias de negociação, TesterHideIndicators, Trabalhando com os resultados da otimização, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

© 2000-2024, MetaQuotes Ltd.

OnTesterInit

É chamada em EAs quando ocorre o evento TesterInit para executar as ações necessárias antes do início da otimização no testador de estratégia. Existem duas versões da função.

Versão com retorno de resultado

```
int  OnTesterInit(void);
```

Valor retornado

Valor do tipo int, zero significa inicialização bem-sucedida do EA em execução no gráfico antes do início da otimização.

O uso da chamada da OnTesterInit() com o retorno de resultado é prioridade, pois este método permite não apenas inicializar o programa, mas também retornar o código de erro em caso de término antecipado da otimização. Retorno de qualquer valor diferente de INIT_SUCCEEDED (0) significa um erro e não será inicializada a otimização.

Versão sem retorno do resultado é deixada apenas para compatibilidade com códigos antigos. Não é recomendada

```
void  OnTesterInit(void);
```

Observação

O evento TesterInit é gerado automaticamente antes do início da otimização do EA no testador de estratégias.Para esse evento, o EA com manipulador OnTesterDeinit() e/ou OnTesterPass() é carregado automaticamente num gráfico de terminal separado com o símbolo e o período especificados no testador.

Esse EA recebe os eventos TesterInit, TesterDeinit e TesterPass, mas, não, os eventos Init, Deinit e NewTick. Assim, toda lógica necessária para processar os resultados de cada passagem durante a otimização deve ser implementada nos manipuladores OnTesterInit(), OnTesterDeinit() e OnTesterPass().

O resultado de cada passagem única ao otimizar uma estratégia pode ser passado por um quadro a partir do manipulador OnTester() com a ajuda da função FrameAdd().

A função OnTesterInit() se destina a inicializar o EA antes de iniciar a otimização para o seguinte processamento de resultados de otimização. É sempre usada em conjunto com o manipulador OnTesterDeinit().

À execução da OnTesterInit() é atribuído um tempo limitado, após o qual o trabalho do EA é forçadamente concluído, e, consequentemente, é cancelada a própria otimização. Nesse caso, uma mensagem é exibida no log do testador:

```
TesterOnTesterInit works too long. Tester cannot be initialized.
```

Exemplo tirado da OnTick, adicionado o manipulador OnTesterInit() para definir os parâmetros de otimização:

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Exemplo de EA com manipulador OnTesterInit()"
#property description "em que são definidos valores e limites "
#property description "de parâmetros de entrada ao otimizar"

input double lots=0. 1;       // volume em lotes
input double kATR=3;          // tamanho da vela de sinal no ATR
input int    ATRperiod=20;    // período do indicador ATR
input int    holdbars=8;      // número de barras para manter a posição
input int    slippage=10;     // derrapagem admissível
input bool   revers=false;    // invertemos o sinal?
input ulong  EXPERT_MAGIC=0;  // MagicNumber do EA
//--- para armazenar o identificador do indicador ATR
int atr_handle;
//--- aqui vamos armazenar os últimos valores do ATR e o corpo da vela
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- lembramo-nos da hora de início da otimização
datetime optimization_start;
//--- para exibir no gráfico a duração após o final da otimização
string report;
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- definimos os valores dos parâmetros de entrada para a otimização
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("Definidos os valores iniciais e limites dos parâmetros de otimização");
//--- lembramo-nos do início da otimização
   optimization_start=TimeLocal();
   report=StringFormat("%s: otimização iniciada em %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- exibimos uma mensagem no gráfico e no log do terminal
   Print(report);
   Comment(report);
//---
  }
```

```
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- duração da otimização
   string log_message=StringFormat("%s: a otimização demorou %d segundos",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- inicializamos as variáveis globais
   last_atr=0;
   last_body=0;
//--- definimos o volume correto
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- criamos o identificador do indicador ATR
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: não foi possível criar o iATR, código de erro %d",__FUNCTION__,
      return(INIT_FAILED);
     }
//--- inicialização bem-sucedida do EA
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- sinal de negociação
   static int signal=0; // +1 significa um sinal de compra, -1 significa um sinal de v
//--- verificamos e fechamos as posições abertas antigas, abertas há mais de holdbars
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- verificamos o surgimento de uma nova barra
   if(isNewBar())
     {
      //--- verificamos a presença de sinal
      signal=CheckSignal();
     }
//--- se aberta uma posição de 'netting', ignoramos o sinal e esperamos até que ele fe
```

© 2000-2024, MetaQuotes Ltd.

```
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // saímos do manipulador de eventos NewTick e não entramos no mercado at
     }
//--- para contas de cobertura ('hedge'), casa posição tem vida e fecha separadamente
   if(signal!=0)
     {
      //--- sinal de compra
      if(signal>0)
        {
         PrintFormat("%s: Existe um sinal de compra! Revers=%s",__FUNCTION__,string(re
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- sinal de venda
      if(signal<0)
        {
         PrintFormat("%s: Existe um sinal de venda! Revers=%s",__FUNCTION__,string(rev
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- fim da função OnTick
  }
//+------------------------------------------------------------------+
//| Verificando se á sinal de negociação                             |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0 significa que não há sinal
   int res=0;
//--- obtemos o valor do ATR na penúltima barra concluída (o índice da barra igual a 2
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- recebemos os dados da última barra fechada numa matriz do tipo MqlRates
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- calculamos o tamanho do corpo da barra na última barra fechada
         last_body=bar[0].close-bar[0].open;
         //--- se o corpo da última barra (com índice 1) exceder o valor anterior do A
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // para a leva altista um valor positivo
        }
      else
         PrintFormat("%s: Não foi possível obter a última barra! Erro",__FUNCTION__,Ge
```

© 2000-2024, MetaQuotes Ltd.

```
      }
    else
       PrintFormat("%s: Não foi possível obter o valor do indicador ATR! Erro",__FUNCTI
//--- se estiver ativado o modo de negociação de reversão
    res=revers?-res:res;  // se necessário, revertemos o sinal (em vez de 1, retornamos
//--- retornamos o valor do sinal de negociação
    return (res);
   }
//+------------------------------------------------------------------+
//|  Retornando true quando aparece uma nova barra                   |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
   {
    static datetime bartime=0; // armazenamos o tempo de abertura da barra atual
//--- obtemos o tempo de abertura da barra zero
    datetime currbar_time=iTime(_Symbol,_Period,0);
//--- se o tempo de abertura mudar, é porque apareceu uma nova barra
    if(bartime!=currbar_time)
      {
       bartime=currbar_time;
       lastbar_timeopen=bartime;
       //--- exibir no log informações sobre o tempo de abertura da nova barra
       if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
         {
          //--- exibimos uma mensagem sobre o tempo de abertura da nova barra
          PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                      StringSubstr(EnumToString(_Period),7),
                      TimeToString(TimeCurrent(),TIME_SECONDS));
          //--- obtemos os dados do último tick
          MqlTick last_tick;
          if(!SymbolInfoTick(Symbol(),last_tick))
             Print("SymbolInfoTick() failed, error = ",GetLastError());
          //--- exibimos o tempo do último tick em segundos
          PrintFormat("Last tick was at %s.%03d",
                      TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
         }
       //--- temos uma nova barra
       return (true);
      }
//--- não há nenhuma barra nova
    return (false);
   }
//+------------------------------------------------------------------+
//| Comprando a mercado com o volume especificado                    |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
   {
//--- compramos a mercado
    return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Vendendo a mercado com o volume definido                         |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- vendemos a mercado
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Fechando posições segundo o tempo de retenção nas barras         |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // número de posições abertas
//--- pesquisa detalhada de todas as posições abertas
   for(int i=total-1; i>=0; i--)
     {
      //--- parâmetros da posição
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- se a posição tem vivido por um longo tempo, e também o MagicNumber e o sím
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // reduzimos o registro do texto para uma correta fo
         PrintFormat("Fechamos a posição #%d %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- definindo o tipo de ordem e de envio do pedido de negociação
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Preparando e enviando uma solicitação de negociação              |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- declaração e inicialização de estruturas
```

```
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- parâmetros da solicitação
   request.action   =TRADE_ACTION_DEAL;                     // tipo de operação de neg
   request.position =pos_ticket;                            // boleta da posição, se i
   request.symbol   =Symbol();                              // símbolo
   request.volume   =volume;                                // volume
   request.type     =type;                                  // tipo de ordem
   request.price    =price;                                 // preço de transação
   request.deviation=slip;                                  // desvio permitido em rel
   request.magic    =magicnumber;                           // MagicNumber da ordem
//--- envio do pedido
   if(!OrderSend(request,result))
     {
      //--- exibimos as informações sobre a falha
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- relatamos sobre a operação bem-sucedida
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Veja também

Teste de estratégias de negociação, Trabalhando com resultados de otimização, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2024, MetaQuotes Ltd.

OnTesterDeinit

É chamada em EAs quando ocorre o evento TesterDeinit para realizar as ações necessárias após a otimização do EA.

```
void  OnTesterDeinit(void);
```

Valor retornado

Sem valor retornado

Observação

O evento TesterDeinit é gerado automaticamente no final da otimização do EA no testador de estratégia.

EA com manipulador OnTesterDeinit() ou OnTesterPass() que é carregado automaticamente num gráfico de terminal separado com o símbolo e o período especificados no testador quando é iniciada a otimização. A função se destina ao processamento final de todos os resultados de otimização.

Deve ser lembrado que os quadros de otimização enviados pelos agentes de teste usando a função FrameAdd() podem vir em pacotes e levar tempo para serem entregues. Portanto, nem todos os quadros e, consequentemente, os eventos TesterPass, podem ser recebidos e processados antes do final da otimização na OnTesterPass(). Portanto, para garantir o recebimento de todos os quadros atrasados na OnTesterDeinit(), você precisa colocar um bloco de código usando a função FrameNext().

Veja também

Teste de estratégias de negociação, Trabalhando com os resultados da otimização, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2024, MetaQuotes Ltd.

OnTesterPass

É chamada em EAs quando ocorre o evento TesterPass para processar o novo quadro de dados durante a otimização do EA.

```
void  OnTesterPass(void);
```

Valor retornado

Sem valor retornado

Observação

O evento TesterPass é gerado automaticamente quando chega um quadro durante a otimização do EA no testador de estratégia.

EA com manipulador OnTesterDeinit() ou OnTesterPass() que é carregado automaticamente num gráfico de terminal separado com o símbolo e o período especificados no testador quando é iniciada a otimização. A função é projetada para processar quadros recebidos de agentes de teste durante a otimização. O envio do quadro com resultados de teste deve ser feito a partir do manipulador OnTester() usando a função FrameAdd().

Deve ser lembrado que os quadros de otimização enviados pelos agentes de teste usando a função FrameAdd() podem vir em pacotes e levar tempo para serem entregues. Portanto, nem todos os quadros e, consequentemente, os eventos TesterPass, podem ser recebidos e processados antes do final da otimização na OnTesterPass(). Portanto, para garantir o recebimento de todos os quadros atrasados na OnTesterDeinit(), você precisa colocar um bloco de código usando a função FrameNext().

Após concluída a otimização de OnTesterDeinit(), você pode classificar novamente todos os quadros com a ajuda das funções FrameFirst()/FrameFilter e FrameNext().

Veja também

Teste de estratégias de negociação, Trabalhando com resultados de otimização, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

© 2000-2024, MetaQuotes Ltd.

Obtendo Informações de Mercado

Estas são funções destinadas a receber informações sobre o estado do mercado.

Função	Ação
SymbolsTotal	Retorna o número de ativos disponíveis (selecionados na janela Observação de Mercado ou todos)
SymbolExist	Verifica se há um símbolo com o nome especificado
SymbolName	Retorna o nome de um ativo especificado
SymbolSelect	Seleciona um ativo na janela Observação de Mercado ou remove um ativo desta janela
SymbolIsSynchronized	Verifica se dados de um ativo selecionado no terminal estão sincronizados com dados no servidor de negociações
SymbolInfoDouble	Retorna um valor double de um ativo para a propriedade correspondente
SymbolInfoInteger	Retorna um valor de tipo inteiro (long, datetime, int ou bool) de um ativo especificado para a propriedade correspondente
SymbolInfoString	Retorna um valor do tipo string de um ativo especificado para a propriedade correspondente
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction
SymbolInfoTick	Retorna os preços correntes para um ativo especificado em uma variável do tipo MqlTick
SymbolInfoSessionQuote	Permite obter a hora de início e fim das sessões de cotação especificadas para um ativo especificado e dia da semana.
SymbolInfoSessionTrade	Permite obter a hora de início e fim das sessões de negociação especificadas para um ativo especificado e dia da semana.
MarketBookAdd	Fornece abertura da Profundidade de Mercado para um ativo selecionado, e subscrever para receber notificações de alterações do DOM (Depth of Market)
MarketBookRelease	Fornece fechamento da Profundidade de Mercado para um ativo selecionado, e cancelar a subscrição para receber notificações de alterações do DOM
MarketBookGet	Retorna um array de estrutura MqlBookInfo contendo registros da Profundidade de Mercado de uma ativo especificado

© 2000-2024, MetaQuotes Ltd.

SymbolsTotal

Retorna o número de ativos disponíveis (selecionados na janela Observação do Mercado ou todos os ativos).

```
int  SymbolsTotal(
   bool  selected      // True - somente ativos em Observador de Mercado
   );
```

Parâmetros

selected

[in] Mode de solicitação. Pode ser true ou false.

Valor do Retorno

Se o parâmetro 'selected' for true, a função retorna o número de ativos selecionados na janela Observador de Mercado. Se o valor for false, ela retorna o número total de todos os ativos.

© 2000-2024, MetaQuotes Ltd.

SymbolExist

Verifica se há um símbolo com o nome especificado.

```
bool  SymbolExist(
   const string  name,     // nome do símbolo
   bool&   is_custom       // tipo de símbolo personalizado
   );
```

Parâmetros

nome

[in] Nome do símbolo.

is_custom

[out] Tipo de símbolo personalizado que é definido após a execução bem-sucedida. Se for true, o símbolo encontrado é personalizado.

Valor retornado

Retorna false se o símbolo não for encontrado entre os padrão ou entre os símbolos personalizados.

Veja também

SymbolsTotal, SymbolSelect, Símbolos personalizados

© 2000-2024, MetaQuotes Ltd.

SymbolName

Retorna o nome de um ativo.

```
string  SymbolName(
   int   pos,          // número na lista
   bool  selected      // true - somente ativos em Observador de Mercado
   );
```

Parâmetros

pos

[in] Número de ordem de um ativo.

selected

[in] Mode de solicitação. Se o valor for true, o ativo é tomado da lista de ativos selecionados na janela Observador de Mercado. Se o valor for false, o ativo é tomado da lista geral.

Valor do Retorno

Valor de tipo string com o nome do ativo.

© 2000-2024, MetaQuotes Ltd.

SymbolSelect

Seleciona um ativo da janela Observador de Mercado ou remove um ativo desta janela.

```
bool  SymbolSelect(
   string  name,        // nome do ativo
   bool    select       // adicionar ou remover
   );
```

Parâmetros

name

[in] Nome do ativo.

select

[in] Alterna. Se o valor for false, o ativo deve ser removido do Observador de Mercado, caso contrário o ativo deve ser selecionado nesta janela. Um ativo não pode ser removido se o gráfico do ativo estiver aberto, ou houver posições em aberto deste ativo.

Valor do Retorno

Em caso de falha retorna false.

© 2000-2024, MetaQuotes Ltd.

SymbolIsSynchronized

A função verifica se os dados de um ativo selecionado no terminal estão sincronizados com os dados no servidor de negociação.

```
bool  SymbolIsSynchronized(
   string  name,        // nome do ativo
   );
```

Parâmetros

name

[in] Nome do ativo.

Valor de retorno

Se os dados estiverem sincronizados, retorna 'true'; caso contrário, retorna 'false'.

Também Veja

SymbolInfoInteger, Organizando Acesso a Dados

© 2000-2024, MetaQuotes Ltd.

SymbolInfoDouble

Retorna a propriedade correspondente de um ativo especificado. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  SymbolInfoDouble(
   string                    name,        // ativo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // identificador da propriedade
   );
```

2. Retorna true ou false dependendo se a função for executada com sucesso. Em caso de sucesso, o valor da propriedade é colocado em uma variável recipiente, passada por referênce atráves do último parâmetro.

```
bool  SymbolInfoDouble(
   string                    name,        // ativo
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // identificador da propriedade
   double&                   double_var   // aqui nós assumimos o valor da propriedade
   );
```

Parâmetros

name

[in] Nome do ativo.

prop_id

[in] Identificador de um propriedade de ativo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_DOUBLE.

double_var

[out] Variável de tipo double recebendo o valor da propriedade solicitada.

Valor do Retorno

Valor de tipo double. Em caso de falha de execução, a informação sobre o erro pode ser obtida usando a função GetLastError():

- 5040 - parâmetro de string inválido para especificar um nome de ativo,
- 4301 - ativo desconhecido (instrumento financeiro),
- 4302 - ativo não está selecionado no "Observador de Mercado" (não encontrado na lista de ativos disponíveis),
- 4303 - identificador inválido de uma propriedade de ativo.

Observação

É recomendável usar SymbolInfoTick() se a função for usada para obter informações sobre o último tick. É bom possível que nenhuma cotação tenha aparecido ainda desde que o terminal se conectou a uma conta de negociação. Em tal situação, o valor solicitado será indefinido

Na maioria dos casos, é suficiente usar a função SymbolInfoTick() permitindo a um usuário receber os valores dos preços Compra, Venda, Último, Volume e a hora da chegada do último tick através de uma única chamada.

© 2000-2024, MetaQuotes Ltd.

A função SymbolInfoMarginRate() fornece informações sobre o valor da margem cobrada, dependendo do tipo e direção da ordem.

Exemplo:

```
void OnTick()
  {
//--- obtém spread a partir das propriedade do ativo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d pontos\r\n",
                            spreadfloat?"flutuante":"fixo",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- agora vamos calcular nós mesmos o spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Spread calculada = "+(string)spread_points+" pontos";
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

SymbolInfoInteger

Retorna a propriedade correspondente de um ativo especificado. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  SymbolInfoInteger(
   string                    name,       // ativo
   ENUM_SYMBOL_INFO_INTEGER  prop_id     // identificador da propriedade

   );
```

2. Retorna true ou false dependendo se a função for executada com sucesso. Em caso de sucesso, o valor da propriedade é colocado em uma variável recipiente, passada por referênce atráves do último parâmetro.

```
bool  SymbolInfoInteger(
   string                    name,       // ativo
   ENUM_SYMBOL_INFO_INTEGER  prop_id,    // identificador da propriedade
   long&                     long_var    // aqui nós assumimos o valor da propriedade
   );
```

Parâmetros

name

[in] Nome do ativo.

prop_id

[in] Identificador de um propriedade de ativo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_INTEGER.

long_var

[out] Variável de tipo long rececendo o valor da propriedade solicitada.

Valor do Retorno

Valor de tipo long. Em caso de falha de execução, a informação sobre o erro pode ser obtida usando a função GetLastError():

- 5040 - parâmetro de string inválido para especificar um nome de ativo,
- 4301 - ativo desconhecido (instrumento financeiro),
- 4302 - ativo não está selecionado no "Observador de Mercado" (não encontrado na lista de ativos disponíveis),
- 4303 - identificador inválido de uma propriedade de ativo.

Observação

É recomendável usar SymbolInfoTick() se a função for usada para obter informações sobre o último tick. É bom possível que nenhuma cotação tenha aparecido ainda desde que o terminal se conectou a uma conta de negociação. Em tal situação, o valor solicitado será indefinido

Na maioria dos casos, é suficiente usar a função SymbolInfoTick() permitindo a um usuário receber os valores dos preços Compra, Venda, Último, Volume e a hora da chegada do último tick através de uma única chamada.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
void OnTick()
  {
//--- obtém spread a partir das propriedade do ativo
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d pontos\r\n",
                            spreadfloat?"flutuante":"fixo",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- agora vamos calcular nós mesmos o spread
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Spread calculada = "+(string)spread_points+" pontos";
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

SymbolInfoString

Retorna a propriedade correspondente de um ativo especificado. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  SymbolInfoString(
   string                    name,         // Ativo
   ENUM_SYMBOL_INFO_STRING   prop_id       // Identificador de propriedade
   );
```

2. Retorna true ou false, dependendo do sucesso na execução da função. Se for bem sucedido, o valor da propriedade é colocada em uma variável de recbimento passada por referência no último parâmetro.

```
bool  SymbolInfoString(
   string                    name,         // Ativo
   ENUM_SYMBOL_INFO_STRING   prop_id,      // Identificador de propriedade
   string&                   string_var    // Aqui nós assumimos o valor da propriedade
   );
```

Parâmetros

name

[in] Nome do ativo.

prop_id

[in] Identificador de um propriedade de ativo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_STRING.

string_var

[out] Variável de tipo string recebendo o valor da propriedade solicitada.

Valor do Retorno

Valor de tipo string. Em caso de falha de execução, a informação sobre o erro pode ser obtida usando a função GetLastError():

- 5040 - parâmetro de string inválido para especificar um nome de ativo,
- 4301 - ativo desconhecido (instrumento financeiro),
- 4302 - ativo não está selecionado no "Observador de Mercado" (não encontrado na lista de ativos disponíveis),
- 4303 - identificador inválido de uma propriedade de ativo.

Observação

É recomendável usar SymbolInfoTick() se a função for usada para obter informações sobre o último tick. É bom possível que nenhuma cotação tenha aparecido ainda desde que o terminal se conectou a uma conta de negociação. Em tal situação, o valor solicitado será indefinido

Na maioria dos casos, é suficiente usar a função SymbolInfoTick() permitindo a um usuário receber os valores dos preços Compra, Venda, Último, Volume e a hora da chegada do último tick através de uma única chamada.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoMarginRate

Retorna uma coleção de índices de margem, dependendo do tipo e direção mandado.

```
bool  SymbolInfoMarginRate(
   string              name,                      // símbolo
   ENUM_ORDER_TYPE     order_type,                // tipo de ordem
   double&             initial_margin_rate,       // taxa de carregamento da margem ini
   double&             maintenance_margin_rate    // taxa de carregamento da margem de m
   );
```

parâmetros

name

[in] Nome do símbolo.

order_type

[in] Tipo de ordem.

initial_margin_rate

[in] Variável do tipo double para carregar o coeficiente de margem inicial. Margem inicial - é o tamanho da quantidade de garantia sob o volume de transação de um lote de direção correspondente. A multiplicação do coeficiente para a margem inicial, pode-se obter o montante de fundos a ser reservados na conta quando se coloca o tipo de ordem especificada.

maintenance_margin_rate

[out] A variável do tipo double para o carregamento de manutenção margem de proporção. Margem de Manutenção - Quantidade mínima para manter um volume de uma posição aberta de uma direção correspondente. A multiplicação do coeficiente da margem de apoio, pode ser obter o montante de fundos a ser reservados na cont, após a atuação do referido tipo de ordem.

valor de retorno

Retorna true se as propriedades da consulta foram bem sucedidas, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoTick

A funcção retorna preços correntes de um ativo especificado em uma variável de tipo MqlTick.

```
bool  SymbolInfoTick(
   string    symbol,      // nome do ativo
   MqlTick&  tick         // referencia a uma estrutura
   );
```

Parâmetros

symbol

[in] Nome do ativo.

tick

[out] Estrutura do tipo MqlTick, na qual os preços correntes e a hora da atualização do último preço será colocada.

Valor do Retorno

A função retorna true se bem sucedida, caso contrário retorna false.

© 2000-2024, MetaQuotes Ltd.

SymbolInfoSessionQuote

Permite obter a hora de início e fim das sessões de cotação especificadas para um ativo especificado e dia da semana.

```
bool  SymbolInfoSessionQuote(
   string            name,              // nome do ativo
   ENUM_DAY_OF_WEEK  day_of_week,       // dia da semana
   uint              session_index,     // índice da sessão
   datetime&         from,              // hora de início da sessão
   datetime&         to                 // hora de término da sessão
   );
```

Parâmetros

name

[in] Nome do ativo.

ENUM_DAY_OF_WEEK

[in] Dia da semana, um valor da enumeração ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de uma sessão, cujo hora de começo e término nós queremos receber. A indexação de sessões começa com 0.

from

[out] Hora de início da sessão em segundos a partir de 00 horas 00 minutos, no valor de retorna a data deve ser ignorada.

to

[out] Hora de término da sessão em segundos a partir de 00 horas e 00 minutos, no valor retornado a data deve ser ignorada.

Valor do Retorno

Se os dados para a sessão, ativo e dia da semana especificados forem recebidos, retorna true, caso contrário retorna false.

Também Veja

Propriedade de Ativo, TimeToStruct, Estrutura de Dados

© 2000-2024, MetaQuotes Ltd.

SymbolInfoSessionTrade

Permite obter a hora de início e fim das sessões de negociação especificadas para um ativo especificado e dia da semana.

```
bool  SymbolInfoSessionTrade(
   string            name,              // nome do ativo
   ENUM_DAY_OF_WEEK  day_of_week,       // dia da semana
   uint              session_index,     // índice da sessão
   datetime&         from,              // hora de início da sessão
   datetime&         to                 // hora de término da sessão
   );
```

Parâmetros

name

[in] Nome do ativo.

ENUM_DAY_OF_WEEK

[in] Dia da semana, um valor da enumeração ENUM_DAY_OF_WEEK.

uint

[in] Número ordinal de uma sessão, cujo hora de começo e término nós queremos receber. A indexação de sessões começa com 0.

from

[out] Hora de início da sessão em segundos a partir de 00 horas 00 minutos, no valor de retorna a data deve ser ignorada.

to

[out] Hora de término da sessão em segundos a partir de 00 horas e 00 minutos, no valor retornado a data deve ser ignorada.

Valor de retorno

Se os dados para a sessão, ativo e dia da semana especificados forem recebidos, retorna true, caso contrário retorna false.

Também Veja

Propriedade de Ativo, TimeToStruct, Estrutura de Dados

© 2000-2024, MetaQuotes Ltd.

MarketBookAdd

Fornece a abertura da Profundidade de Mercado (DOM) para um ativo selecionado, e subscreve para receber notificados de alterações na DOM (Depth of Market).

```
bool  MarketBookAdd(
   string  symbol      // ativo
   );
```

Parâmetros

symbol

[in] O nome de um ativo, cuja Profundidade de Mercado é para ser usada no Expert Advisor ou script.

Valor do Retorno

O valor true se aberto com sucesso, caso contrário false.

Observação

Normalmente, esta função deve ser chamada a partir da função OnInit() ou no construtor da classe. Para manipular a chegada de alertas, o programa Expert Advisor deve conter a função void OnBookEvent(string& symbol).

Também Veja

Estrutura da Profunção de Mercado, Estruturas e Classes

© 2000-2024, MetaQuotes Ltd.

MarketBookRelease

Fornece o fechamento da Profundidade de Mercado (DOM) para um ativo selecionado, e cancela a subscrição para receber notificações de alteração na DOM (Depth of Market).

```
bool  MarketBookRelease(
   string  symbol      // ativo
   );
```

Parâmetros

symbol

[in] Nome do ativo.

Valor do Retorno

O valor true se a fechamento foi bem sucedido, caso contrário false.

Observação

Normalmente, esta função deve ser chamada da função OnDeinit(), se a correspondente função MarketBookAdd() foi chamada na função OnInit(). Ou ela deve ser chamada a partir do destrutor da classe, se a correspondente função MarketBookAdd() foi chamada a partir do construtor da classe.

Também Veja

Estrutura da Profundidade de Mercado, Estruturas e Classes

© 2000-2024, MetaQuotes Ltd.

MarketBookGet

Retorna um array de estruturas MqlBookInfo contendo registros da Profundidade de Mercado de um ativo especificado.

```
bool  MarketBookGet(
   string        symbol,      // ativo
   MqlBookInfo&  book[]       // referência para um array
   );
```

Parâmetros

symbol

[in] Nome do ativo.

book[]

[in] Referência para um array de registros de Profundidade de Mercado. O array pode ser pré-alocado para um número suficiente de registros. Se um array dinâmico não tiver sido pré-alocado na memória operacional, o terminal cliente alocará ele mesmo o array.

Valor do Retorno

Retorna true em caso de sucesso, caso contrário false.

Observação

A Profundidade de Mercado deve ser pre-aberta pela função MarketBookAdd().

Exemplo:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo para ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " tipo = ",priceArray[i].type);
        }
     }
   else
     {
      Print("Could not get contents of the symbol DOM ",Symbol());
     }
```

Também Veja

Estrutura da Profundidade de Mercado, Estruturas e Classes

© 2000-2024, MetaQuotes Ltd.

Funções do Calendário Econômico

Esta seção descreve as funções para trabalhar com o Calendário econômico que está disponível diretamente na plataforma MetaTrader. O calendário econômico é uma enciclopédia pronta com descrição, data de lançamento e grau de importância de indicadores macroeconômicos. Os valores reais dos indicadores macroeconômicos entram na plataforma MetaTrader imediatamente no momento da publicação e são exibidos no gráfico como tags, permitindo monitorar visualmente os indicadores necessários por países, por moedas e por importância.

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

As funções do calendário econômico permitem analisar automaticamente os eventos recebidos de acordo com critérios de importância individuais - países/moedas.

Função	Ação
CalendarCountryById	Obtém uma descrição do país por seu ID
CalendarEventById	Obtém a descrição do evento por seu ID
CalendarValueById	Obtém uma descrição do valor do evento por seu ID
CalendarCountries	Obtém um array de descrições dos países disponíveis no calendário.
CalendarEventByCountry	Obtém um array de descrições de todos os eventos disponíveis no Calendário, segundo o código do país especificado
CalendarEventByCurrency	Obtém um array de descrições de todos os eventos disponíveis no calendário, de acordo com a moeda especificada
CalendarValueHistoryByEvent	Obtém um array de valores de todos os eventos no intervalo de tempo especificado, segundo o ID do evento
CalendarValueHistory	Obtém uma matriz de valores de todos os eventos num determinado intervalo de tempo com um filtro por país e/ou moeda
CalendarValueLastByEvent	Obtém um array de valores do eventos por seu ID desde o momento em que criado do banco de dados do Calendário com o change_id definido
CalendarValueLast	Obtém um array de valores de todos os eventos com filtro por país e/ou moeda desde o momento em que criado do banco de dados do Calendário com o change_id definido

© 2000-2024, MetaQuotes Ltd.

CalendarCountryById

Obtém a descrição do país por seu ID.

```
bool  CalendarCountryById(
   const long           country_id,      // identificador do país
   MqlCalendarCountry&  country          // variável para obter a descrição do país
   );
```

Parâmetros

country_id

[in] ID do país segundo o padrão ISO 3166-1.

country

[out] Variável do tipo MqlCalendarCountry para obter uma descrição do país.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 5402 - ERR_CALENDAR_NO_DATA (país não encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite de solicitação excedido),

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenha a lista de países do Calendário Econômico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- verifique o resultado
   if(count==0)
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
//--- se você tiver dois ou mais países
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- agora obtenha a descrição do país pelo seu identificador
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- prepare a descrição do país
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("name = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("currency = " + country.currency+"\n");
         descr+=("currency_symbol = " + country.currency_symbol+"\n");
```

© 2000-2024, MetaQuotes Ltd.

```
         descr+=("url_name = " + country.url_name);
         //--- imprima a descrição do país
         Print(descr);
        }
      else
         Print("CalendarCountryById() failed. Error ",GetLastError());
     }
//---
  }
/*
  Resultado:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

Veja também

CalendarCountries, CalendarEventByCountry

© 2000-2024, MetaQuotes Ltd.

CalendarEventById

Obtém uma descrição do evento por seu ID.

```
bool  CalendarEventById(
   ulong                event_id,      // ID do evento
   MqlCalendarEvent&    event          // variable para obter a descrição do evento
   );
```

Parâmetros

event_id

[in] ID do evento.

event

[out] Variável de tipo MqlCalendarEvent para obter uma descrição do evento.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 5402 - ERR_CALENDAR_NO_DATA (país não encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite de solicitação excedido),

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código do país para a Alemanha de acordo com a ISO 3166-1 Alpha-2
   string germany_code="DE";
//--- obtenha os eventos para a Alemanha
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- imprima os eventos para a Alemanha no Diário
   if(events_count>0)
     {
      PrintFormat("Eventos para a Alemanha: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("Falha ao obter eventos para o código do país %s, erro %d",
                  germany_code,GetLastError());
      //--- conclusão antecipada do script
      return;
     }
//--- obtenha a descrição do último evento do array events[]
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("Obtida a descrição do evento com event_id=%d",event_id);
      PrintFormat("País: %s (código do país= %d)",country.name,event.country_id);
      PrintFormat("Nome do evento: %s",event.name);
      PrintFormat("Código do evento: %s",event.event_code);
      PrintFormat("Importância do evento: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPORT
      PrintFormat("Tipo de evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.ty
      PrintFormat("Setor do evento: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)event
      PrintFormat("Frequência do evento: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUEN
      PrintFormat("Modo de saída do evento: %s",EnumToString((ENUM_CALENDAR_EVENT_TIME
      PrintFormat("Unidade de medida do evento: %s",EnumToString((ENUM_CALENDAR_EVENT_
      PrintFormat("Número de casas decimais: %d",event.digits);
      PrintFormat("Multiplicador de valor: %s",EnumToString((ENUM_CALENDAR_EVENT_MULTI
      PrintFormat("URL da fonte: %s",event.source_url);
     }
   else
      PrintFormat("Falha ao obter a descrição do evento para event_d=%s, erro %d",
                  event_id,GetLastError());
  }
/*
  Resultado:
  Eventos para a Alemanha: 50
             [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importa
   [ 0] 276010001      1        6           2           0          276      1
   [ 1] 276010002      1        6           2           0          276      1
   [ 2] 276010003      1        4           2           0          276      1
   [ 3] 276010004      1        4           2           0          276      1
   ....
   [47] 276500001      1        8           2           0          276      0
   [48] 276500002      1        8           2           0          276      0
   [49] 276500003      1        8           2           0          276      0
  Descrições de evento recebidas com event_id=276500003
  País: Germany (código do país = 276)
  Nome do evento: Markit Composite PMI
  Código do evento: markit-composite-pmi
   Importância do evento: CALENDAR_IMPORTANCE_MODERATE
   Tipo de evento: CALENDAR_TYPE_INDICATOR
   Setor do evento: CALENDAR_SECTOR_BUSINESS
   Frequência do evento: CALENDAR_FREQUENCY_MONTH
   Modo de saída do evento: CALENDAR_TIMEMODE_DATETIME
   Unidade de medida do evento: CALENDAR_UNIT_NONE
  Número de casas decimais: 1
   Multiplicador de valor: CALENDAR_MULTIPLIER_NONE
   URL da fonte: https://www.markiteconomics.com
```

© 2000-2024, MetaQuotes Ltd.

```
*/
```

Veja também

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueById

Obtém uma descrição do valor do evento por seu ID.

```
bool  CalendarValueById(
   ulong                value_id,      // ID do valor do evento
   MqlCalendarValue&    value          // variável para obter o valor do evento
   );
```

Parâmetros

value_id

[in] Identificador do valor do evento.

value

[out] Variável de tipo MqlCalendarValue para obter o valor do evento. Veja um exemplo de processamento de eventos de calendário.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 5402 - ERR_CALENDAR_NO_DATA (país não encontrado),
- 5401 - ERR_CALENDAR_TIMEOUT (tempo limite de solicitação excedido)

Observação

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos actual_value, forecast_value, prev_value e revised_prev_value. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código do país para o Japão de acordo com a ISO 3166-1 Alpha-2
```

© 2000-2024, MetaQuotes Ltd.

```
   string japan_code="JP";
//--- defina os limites do intervalo de onde são tomados os eventos
   datetime date_from=D'01.01.2018';  // defina todos os eventos de 2018
   datetime date_to=0;                // 0 indica que todos os eventos são conhecidos
//--- obtenha uma matriz de valores de eventos para o Japão
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- passe pelos valores do evento encontrado
   if(values_count>0)
     {
      PrintFormat("Número de valores para eventos do Japão: %d",values_count);
      //--- remova todos os valores "vazios" (actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Número de valores após a exclusão de vazios: %d",ArraySize(values))
     }
   else
     {
      PrintFormat("Falha ao obter eventos para o código do país %s, erro %d",
                  japan_code,GetLastError());
      //--- conclusão antecipada do script
      return;
     }
//--- não deixe mais de 10 valores no array values[]
   if(ArraySize(values)>10)
     {
      PrintFormat("Reduza a lista de valores para 10 e exiba-a");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- agora exiba como, com base no value_id conhecido, obtém-se uma descrição do valo
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d value=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_I
     }
//---
  }
/*
  Resultado:
  Número de valores para eventos do Japão: 1734
  Número de valores após a exclusão de vazios: 1017
  Reduza a lista de valores para 10 e exiba-a
```

© 2000-2024, MetaQuotes Ltd.

```
         [id] [event_id]              [time]            [period] [revision] [actual_val
    [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0         90
    [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0         70
    [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0        110
    [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0        230
    [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0        163
    [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0         40
    [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0       -180
    [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0         20
    [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1        140
    [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1       -100
  Exiba informações resumidas sobre valores de eventos com base em value_id
   0: value_id=56500 value=900000 impact=CALENDAR_IMPACT_POSITIVE
   1: value_id=56501 value=700000 impact=CALENDAR_IMPACT_NA
   2: value_id=56502 value=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3: value_id=56544 value=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4: value_id=56556 value=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5: value_id=55887 value=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6: value_id=55888 value=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7: value_id=55889 value=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8: value_id=55948 value=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9: value_id=55949 value=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

Veja também

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

© 2000-2024, MetaQuotes Ltd.

CalendarCountries

Obtém um array de descrições dos países disponíveis no calendário.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // array para obter uma lista de descrições
   );
```

Parâmetros

countries[]

[out] Array de tipo MqlCalendarCountry para obter descrições de todos os países do Calendário.

Valor retornado

Número de descrições recebidas. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- 5400 - ERR_CALENDAR_MORE_DATA (tamanho de array insuficiente para obter descrições de todos os países, portanto, eles recebem apenas o que foi incluído).

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtenha a lista de países do Calendário Econômico
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- Imprima o array no Diário
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
/*
  Resultado:
        [id]            [name] [code] [currency] [currency_symbol]        [url_name] [re
   [ 0]    0 "Worldwide"      "WW"   "ALL"      ""                "worldwide"
   [ 1]  999 "European Union" "EU"   "EUR"      "€"               "european-union"
   [ 2]  840 "United States"  "US"   "USD"      "$"               "united-states"
   [ 3]  124 "Canada"         "CA"   "CAD"      "$"               "canada"
   [ 4]   36 "Australia"      "AU"   "AUD"      "$"               "australia"
   [ 5]  554 "New Zealand"    "NZ"   "NZD"      "$"               "new-zealand"
   [ 6]  392 "Japan"          "JP"   "JPY"      "Ɨ"               "japan"
   [ 7]  156 "China"          "CN"   "CNY"      "Ɨ"               "china"
   [ 8]  826 "United Kingdom" "GB"   "GBP"      "J"               "united-kingdom"
   [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"               "switzerland"
```

© 2000-2024, MetaQuotes Ltd.

```
   [10]  276 "Germany"        "DE"   "EUR"      "€"                "germany"
   [11]  250 "France"         "FR"   "EUR"      "€"                "france"
   [12]  380 "Italy"          "IT"   "EUR"      "€"                "italy"
   [13]  724 "Spain"          "ES"   "EUR"      "€"                "spain"
   [14]   76 "Brazil"         "BR"   "BRL"      "R$"               "brazil"
   [15]  410 "South Korea"    "KR"   "KRW"      "₩"                "south-korea"
*/
  }
```

Veja também

CalendarEventByCountry, CalendarCountryById

© 2000-2024, MetaQuotes Ltd.

CalendarEventByCountry

Obtém um array de descrições de todos os eventos disponíveis no Calendário, de acordo com o código do país especificado.

```
int  CalendarEventByCountry(
   string               country_code,     // código do país segundo ISO 3166-1 alpha-2
   MqlCalendarEvent&    events[]          // variável para obter o array de descrições
   );
```

Parâmetros

country_code

[in] Código do país segundo ISO 3166-1 alpha-2

events[]

[out] Array do tipo MqlCalendarEvent para obter descrições de todos os eventos para o país especificado.

Valor retornado

Número de descrições recebidas. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- falha ao executar ArrayResize()

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código do país para a União Europeia de acordo com a ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenha os eventos para a União Europeia
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- imprima eventos para a União Europeia no Diário
   if(events_count>0)
     {
      PrintFormat("Eventos para a União Europeia: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  Resultado:
  Eventos para a União Europeia: 56
```

© 2000-2024, MetaQuotes Ltd.

```
              [id] [type]  [country_id] [unit] [importance] [multiplier] [digits] [ever
   [ 0] 999010001      0          999      0            2            0        0  "ECB
   [ 1] 999010002      0          999      0            2            0        0  "ECB
   [ 2] 999010003      0          999      0            3            0        0  "ECB
   [ 3] 999010004      0          999      0            3            0        0  "ECB
   [ 4] 999010005      0          999      0            2            0        0  "ECB
   [ 5] 999010006      1          999      1            3            0        2  "ECB
   [ 6] 999010007      1          999      1            3            0        2  "ECB
   [ 7] 999010008      0          999      0            2            0        0  "ECB
   [ 8] 999010009      1          999      2            2            3        3  "ECB
   [ 9] 999010010      0          999      0            2            0        0  "ECB
   [10] 999010011      0          999      0            2            0        0  "ECB
   ...

*/
```

Veja também

CalendarCountries, CalendarCountryById

© 2000-2024, MetaQuotes Ltd.

CalendarEventByCurrency

Obtém um array de descrições de todos os eventos disponíveis no Calendário para a moeda especificada.

```
int  CalendarEventByCurrency(
   const string         currency,      // código da moeda do país
   MqlCalendarEvent&    events[]       // variável para obter o array de descrições
   );
```

Parâmetros

currency

[in] Código da moeda do país

events[]

[out] Array do tipo MqlCalendarEvent para obter descrições de todos os eventos da moeda especificada.

Valor retornado

Número de descrições recebidas. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- falha ao executar ArrayResize()

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declare um array para receber eventos do Calendário Econômico
      MqlCalendarEvent events[];
//--- obtenha eventos para a moeda da União Europeia
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- por exemplo, 10 eventos são suficientes para nós
      if(count>10)
         ArrayResize(events,10);
//--- imprima eventos no Diário
      ArrayPrint(events);
  }
/*
  Resultado:
              [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/h
```

© 2000-2024, MetaQuotes Ltd.

```
   [1] 999010002      0          999      0            2 "https://www.ecb.europa.eu/h
   [2] 999010003      0          999      0            3 "https://www.ecb.europa.eu/h
   [3] 999010004      0          999      0            3 "https://www.ecb.europa.eu/h
   [4] 999010005      0          999      0            2 "https://www.ecb.europa.eu/h
   [5] 999010006      1          999      1            3 "https://www.ecb.europa.eu/h
   [6] 999010007      1          999      1            3 "https://www.ecb.europa.eu/h
   [7] 999010008      0          999      0            2 "https://www.ecb.europa.eu/h
   [8] 999010009      1          999      2            2 "https://www.ecb.europa.eu/h
   [9] 999010010      0          999      0            2 "https://www.ecb.europa.eu/h
*/
```

Veja também

CalendarEventById, CalendarEventByCountry

© 2000-2024, MetaQuotes Ltd.

CalendarValueHistoryByEvent

Obtém um array de valores de todos os eventos no intervalo de tempo especificado, segundo o ID do evento.

```
int  CalendarValueHistoryByEvent(
   ulong              event_id,          // identificador do evento
   MqlCalendarValue&  values[],          // array para obter a descrição dos valores
   datetime           datetime_from,     // borda esquerda do intervalo de tempo
   datetime           datetime_to=0      // borda direita do intervalo de tempo
   );
```

Parâmetros

event_id

[in] ID do evento.

values[]

[out] Matriz de tipo MqlCalendarValue para obter valores de eventos. Veja um exemplo de processamento de eventos de calendário.

datetime_from

[in] Data de início do intervalo de tempo a partir do qual os eventos são selecionados pelo ID definido, enquanto *datetime_from* < *datetime_to*.

datetime_to=0

[in] Data final do intervalo de tempo a partir do qual os eventos são selecionados pelo ID definido. Se não for especificado o parâmetro *datetime_to* (ou é igual a 0), serão retornados todos os valores de evento da data especificada *datetime_from*, disponível no banco de dados do Calendário, incluindo os valores dos eventos planejados para o futuro.

Valor retornado

Retorna true em caso de sucesso, caso contrário, fals. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- 5400 - ERR_CALENDAR_MORE_DATA (tamanho de array insuficiente para obter descrições de todos os valores, portanto, eles recebem apenas o que contêm).
- falha ao executar ArrayResize()

Se, para o valor de evento, não houver algum dos campos abaixo

```
struct MqlCalendarValue
  {
   ...
   long              actual_value;              // valor atual do indicador
   long              prev_value;                // valor anterior do indicador
   long              revised_prev_value;        // valor do evento anterior revi
   long              forecast_value;            // valor previsto do evento
   ...
```

© 2000-2024, MetaQuotes Ltd.

```
      };
```

o valor do campo ausente será retornado como INT64_MIN (-9223372036854775808). Veja o exemplo abaixo para valores de campo *revised_prev_value*.

Observação

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos actual_value, forecast_value, prev_value e revised_prev_value. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código do país para a União Europeia de acordo com a ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenha os eventos para a União Europeia
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- imprima eventos para a União Europeia no Diário
   if(events_count>0)
     {
      PrintFormat("Eventos para a União Europeia: %d",events_count);
      //--- reduza a lista de eventos, para estudar basta deixar 10
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- veja que o evento "ECB Interest Rate Decision" tem event_id=999010007
   ulong event_id=events[6].id;        // o ID deste evento pode mudar no Calendário,
   string event_name=events[6].name;   // nome do evento no Calendário
   PrintFormat("Obtenha o valor para o evento event_name=%s event_id=%d",event_name,ev
//--- obtenha todos os valores do evento "ECB Interest Rate Decision"
   MqlCalendarValue values[];
//--- defina os limites do intervalo de onde são tomados os eventos
   datetime date_from=0;           // pegue todos os eventos desde o começo do histór
```

```
   datetime date_to=D'01.01.2016'; // pege eventos não mais antigos que 2016
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("Obtidos valores de eventos %s: %d",
                  event_name,ArraySize(values));
      //--- reduza a lista de eventos para 10, isso chega 10
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Erro! Falha ao obter valores para o evento event_id=%d",event_id);
      PrintFormat("Código de erro: %d",GetLastError());
     }
  }
//---
/*
  Resultado:
  Eventos para a União Europeia: 56
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 999010001      0        5           0           0          999      0
   [1] 999010002      0        5           0           0          999      0
   [2] 999010003      0        5           0           0          999      0
   [3] 999010004      0        5           0           0          999      0
   [4] 999010005      0        5           0           0          999      0
   [5] 999010006      1        5           0           0          999      1
   [6] 999010007      1        5           0           0          999      1
   [7] 999010008      0        5           0           0          999      0
   [8] 999010009      1        5           0           0          999      2
   [9] 999010010      0        5           0           0          999      0
  Obtenha os valores para o evento event_name=ECB Interest Rate Decision event_id=999(
  Obtidos os valores para o evento ECB Interest Rate Decision: 102
       [id] [event_id]              [time]            [period] [revision] [actual_valu
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0        3750(
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0        3750(
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0        4000(
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0        4000(
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0        4000(
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0        4000(
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0        4000(
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0        4000(
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0        4000(
*/
```

Veja também

© 2000-2024, MetaQuotes Ltd.

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueHistory

Obtém uma matriz de valores de todos os eventos num determinado intervalo de tempo com um filtro por país e/ou moeda.

```
int  CalendarValueHistory(
   MqlCalendarValue&  values[],              // array para obter descrições de valores
   datetime           datetime_from,         // borda esquerda do intervalo de tempo
   datetime           datetime_to=0          // borda direita do intervalo de tempo
   const string       country_code=NULL,     // código do país ISO 3166-1 alpha-2
   const string       currency=NULL          // código da moeda do país
   );
```

Parâmetros

values[]

[out] Matriz de tipo MqlCalendarValue para obter valores de eventos. Veja um exemplo de processamento de eventos de calendário.

datetime_from

[in] Data de início do intervalo de tempo a partir do qual os eventos são selecionados pelo ID definido, enquanto *datetime_from* < *datetime_to*.

datetime_to=0

[in] Data final do intervalo de tempo a partir do qual os eventos são selecionados pelo ID definido. Se não for especificado o parâmetro *datetime_to* (ou é igual a 0), serão retornados todos os valores de evento da data especificada *datetime_from*, disponível no banco de dados do Calendário, incluindo os valores dos eventos planejados para o futuro.

country_code=NULL

[in] Código do país segundo ISO 3166-1 alpha-2

currency=NULL

[in] Código da moeda do país

Valor retornado

Se bem-sucedido, retorna o número de valores disponíveis no array de valores, caso contrário -1.. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- 5400 - ERR_CALENDAR_MORE_DATA (tamanho de array insuficiente para obter descrições de todos os valores, portanto, eles recebem apenas o que contêm).
- falha ao executar ArrayResize()

Observação

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os

© 2000-2024, MetaQuotes Ltd.

parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

Se um array foi passado para a função *events[]* de comprimento fixo e, como resultado da solicitação, não havia espaço suficiente para salvar o resultado inteiro, é mostrado o erro ERR_CALENDAR_MORE_DATA (5400).

Se não for especificado o parâmetro *datetime_to* (=0), serão retornados todos os valores de eventos a partir da data *datetime_from* disponível no banco de dados do Calendário, incluindo os valores planejados para o futuro.

Para os filtros *country_code* e *currency* valores NULL e "" são equivalentes, eles indicam que não há filtro.

Para *country_code* deve ser usado o campo *code* da estrutura MqlCalendarCountry, por exemplo, "US", "RU" ou "EU".

Para *currency* deve ser usado o campo *currency* da estrutura MqlCalendarCountry, por exemplo, "USD", "RUB" ou "EUR".

Os filtros são aplicados por conjunção, ou seja, através do booliano 'E' são selecionados apenas os eventos para os quais ambas as condições são atendidas simultaneamente - país e moeda.

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos actual_value, forecast_value, prev_value e revised_prev_value. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- código do país para a União Europeia de acordo com a ISO 3166-1 Alpha-2
   string EU_code="EU";
//--- obtenha todos os valores de evento da União Europeia
   MqlCalendarValue values[];
//--- defina os limites do intervalo de onde são tomados os eventos
   datetime date_from=D'01.01.2018';  // defina todos os eventos de 2018
   datetime date_to=0;                // 0 indica que todos os eventos são conhecidos
/ --- solicite o histórico do evento para a União Europeia desde 2018
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("Obtidos valores do evento para country_code=%s: %d",
```

© 2000-2024, MetaQuotes Ltd.

```
                    EU_code,ArraySize(values));
      //--- reduza o tamanho do array para imprimir no Diário
      ArrayResize(values,10);
//--- imprima o valor dos eventos no Diário
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Erro! Falha ao obter eventos do país country_code=%s",EU_code);
      PrintFormat("Código de erro: %d",GetLastError());
     }
//---
  }
/*
  Resultado:
  Obtidos os valores de eventos para country_code=EU: 1384
        [id] [event_id]           [time]               [period] [revision]   [actual_v
   [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00          3       60600
   [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00          3       56600
   [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00          3       58100
   [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00          0         600
   [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00          0        2800
   [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00          1         900
   [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00          1        1400
   [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00          0      127900
   [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00          0        9100
   [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00          0       18400
*/
```

Veja também

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueLastByEvent

Obtém um array de valores do eventos por seu ID desde o momento em que criado do banco de dados do Calendário com o change_id definido.

```
int  CalendarValueLastByEvent(
   ulong                 event_id,       // identificador de evento
   ulong&                change_id,      // identificador do valor do evento
   MqlCalendarValue&     values[]        // array para obter descrições de valores
   );
```

Parâmetros

event_id

[in] ID do evento.

change_id

[in][out] Identificador de alteração.

values[]

[out] Array de tipo MqlCalendarValue para obter valores de eventos. Veja um exemplo de processamento de eventos de calendário.

Valor retornado

Número de valores de eventos recebidos. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- 5400 - ERR_CALENDAR_MORE_DATA (tamanho de array insuficiente para obter descrições de todos os valores, portanto, eles recebem apenas o que contêm).
- falha ao executar ArrayResize()

Observação

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

Se um array foi passado para a função *events[]* de comprimento fixo e, como resultado da solicitação, não havia espaço suficiente para salvar o resultado inteiro, é mostrado o erro ERR_CALENDAR_MORE_DATA (5400).

Se na função for passado *change_id* = 0, a função sempre retornará zero, mas ao mesmo tempo *change_id* retorna o estado atual do banco de dados do Calendário.

A função retorna uma matriz de valores para as notícias especificadas e *change_id* novo que pode ser usado para chamadas subsequentes desta função para obter novos valores de notícias. Assim, chamando esta função com o último conhecido *change_id*, pode-se receber atualizações de valores para uma determinada notícia.

© 2000-2024, MetaQuotes Ltd.

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos actual_value, forecast_value, prev_value e revised_prev_value. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Um exemplo de um EA que ouve a publicação do Nonfarm payrolls:

```
#property description "Exemplo de uso da função CalendarValueLastByEvent"
#property description " para pegar a publicação do relatório do evento Nonfarm Payroll
#property description "Para fazer isso, você precisa obter o ID atual da alteração"
#property description " base do Calendário. Em seguida, obter segundo este identificac
#property description " apenas novos eventos através de pesquisa no temporizador"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy timer
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- ID da alteração da base do Calendário
   static ulong calendar_change_id=0;
//--- sinal da primeira inicialização
   static bool first=true;
//--- identificador de evento
   static ulong event_id=0;
//--- nome do evento
   static string event_name=NULL;
//--- array de valores do evento
   MqlCalendarValue values[];
//--- inicialize - obtenha o calendar_change_id atual
   if(first)
     {
      MqlCalendarEvent events[];
      //--- código do país para os EUA de acordo com a ISO 3166-1 Alpha-2
      string USA_code="US";
      //--- obtenha eventos para os EUA
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- posição necessário no array de eventos events
      int event_pos=-1;
      //--- registre os eventos para os EUA no Diário
      if(events_count>0)
        {
         PrintFormat("%s: Eventos para os EUA: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- insira o nome do evento em minúsculas
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() retornou o erro %d",GetLastError());
               //--- antecipadamente saia da função
               return;
              }
            //--- procure o evento "Nonfarm Payrolls"
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- evento encontrado, lembre-se de seu id
               event_id=events[i].id;
               //--- escreva para o evento "Nonfarm Payrolls" seu nome
               event_name=events[i].name;
               //--- lembre a posição do evento no array events[]
               event_pos=i;
               //--- na verdade, no Calendário há vários eventos contendo o nome "non
               PrintFormat("Evento \"Nonfarm Payrolls\" encontrado: event_id=%d  event
               //--- veja todos os eventos, comente o operador break, para entender me
               break;
              }
           }
```

```
        //--- reduza a lista, apague os eventos após o evento"Nonfarm Payrolls"
        ArrayRemove(events,event_pos+1);
        //--- para facilitar o estudo, deixe 9 eventos antes de "Nonfarm Payrolls"
        ArrayRemove(events,0,event_pos-9);
        ArrayPrint(events);
       }
     else
       {
        PrintFormat("%s: CalendarEventByCountry(%s) retornou 0 eventos, código de er
                    USA_code,__FUNCTION__,GetLastError());
        //--- falha, tente novamente na próxima chamada do temporizador
        return;
       }

     //--- obtenha, para o evento definido, o ID da alteração da base do Calendário
     if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
       {
        //--- este bloco de código não pode ser executado na primeira inicialização,
        PrintFormat("%s: Obtido ID atual da base do calendário: change_id=%d",
                    __FUNCTION__,calendar_change_id);
        //--- defina o sinalizador e saia antes do próximo evento do temporizador
        first=false;
        return;
       }
     else
       {
        //--- dados não recebidos (para a primeira execução, isso é normal), verifiq
        int error_code=GetLastError();
        if(error_code==0)
          {
           PrintFormat("%s: Obtido ID atual da base do calendário: change_id=%d",
                       __FUNCTION__,calendar_change_id);
           //--- defina o sinalizador e saia antes do próximo evento do temporizador
           first=false;
           //--- agora você tem o valor calendar_change_id
           return;
          }
        else
          {
           //--- isso é realmente um erro
           PrintFormat("%s: Falha ao obter valores para o evento event_id=%d",__FUNC
           PrintFormat("Código de erro: %d",error_code);
           //--- falha, tente novamente na próxima chamada do temporizador
           return;
          }
       }
    }

//--- você tem o último valor conhecido do ID de mudança do calendário (change_id)
```

© 2000-2024, MetaQuotes Ltd.

```
   ulong old_change_id=calendar_change_id;
//--- verifique se surgiu um novo valor para o evento "Nonfarm Payrolls"
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: Obtidos novos eventos para \"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- exiba no Diário informações do array values
      ArrayPrint(values);
      //--- exiba no Diário os valores do ID do Calendário - o anterior e o novo
      PrintFormat("%s: Anterior change_id=%d, novo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      escreva aqui o código que processará a publicação de dados sobre o "Nonfarm Payr
      */
     }
//---
  }
/*
  Resultado:
   OnTimer: Eventos para EUA: 202
   Evento "Nonfarm Payrolls" encontrado: event_id=840030016  event_name=Nonfarm Payrol
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 840030007      1        4           2           0          840      1
   [1] 840030008      1        4           2           0          840      1
   [2] 840030009      1        4           2           0          840      0
   [3] 840030010      1        4           2           0          840      0
   [4] 840030011      1        4           2           0          840      1
   [5] 840030012      1        4           2           0          840      1
   [6] 840030013      1        4           2           0          840      1
   [7] 840030014      1        4           2           0          840      1
   [8] 840030015      1        3           2           0          840      1
   [9] 840030016      1        3           2           0          840      4
   OnTimer: Recebido o ID atual da base do Calendário: change_id=33986560

*/
```

Veja também

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

CalendarValueLast

Obtém um array de valores de todos os eventos com filtro por país e/ou moeda desde o momento em que criado do banco de dados do Calendário com o change_id definido.

```
int  CalendarValueLast(
   ulong&              change_id,            // identificador de alteração
   MqlCalendarValue&   values[],             // array para obter descrições de valo
   const string        country_code=NULL,    // código do país segundo ISO 3166-1 a
   const string        currency=NULL         // código da moeda do país
   );
```

Parâmetros

change_id

[in][out] Identificador de alteração.

values[]

[out] Array de tipo MqlCalendarValue para obter valores de eventos. Veja um exemplo de processamento de eventos de calendário.

country_code=NULL

[in] Código do país segundo ISO 3166-1 alpha-2

currency=NULL

[in] Código da moeda do país

Valor retornado

Número de valores de eventos recebidos. Para obter informações de erro, chame a função GetLastError(). Possíveis erros:

- 4001 - ERR_INTERNAL_ERROR (erro geral de tempo de execução),
- 4004 - ERR_NOT_ENOUGH_MEMORY (memória insuficiente para execução da solicitação),
- 5401 - ERR_CALENDAR_TIMEOUT (excedido tempo limite de solicitação),
- 5400 - ERR_CALENDAR_MORE_DATA (tamanho de array insuficiente para obter descrições de todos os valores, portanto, eles recebem apenas o que contêm).
- falha ao executar ArrayResize()

Observação

Todas as funções para trabalhar com o Calendário Econômico usam o horário do servidor de negociação (TimeTradeServer). Isso significa que o tempo na estrutura MqlCalendarValue e os parâmetros de tempo de entrada nas funções CalendarValueHistoryByEvent/CalendarValueHistory são definidos no fuso horário do servidor de negociação e não no horário local do usuário.

Se um array foi passado para a função *events[]* de comprimento fixo e, como resultado da solicitação, não havia espaço suficiente para salvar o resultado inteiro, é mostrado o erro ERR_CALENDAR_MORE_DATA (5400).

Se na função for passado *change_id* = 0, a função sempre retornará zero, mas ao mesmo tempo *change_id* retorna o estado atual do banco de dados do Calendário.

© 2000-2024, MetaQuotes Ltd.

Para os filtros *country_code* e *currency* valores NULL e "" são equivalentes, eles indicam que não há filtro.

Para *country_code* deve ser usado o campo *code* da estrutura MqlCalendarCountry, por exemplo, "US", "RU" ou "EU".

Para *currency* deve ser usado o campo *currency* da estrutura MqlCalendarCountry, por exemplo, "USD", "RUB" ou "EUR".

Os filtros são aplicados por conjunção, ou seja, através do booliano 'E' são selecionados apenas os eventos para os quais ambas as condições são atendidas simultaneamente - país e moeda

A função retorna uma matriz de valores para as notícias especificadas e *change_id* novo que pode ser usado para chamadas subsequentes desta função para obter novos valores de notícias. Assim, chamando esta função com o último conhecido *change_id*, pode-se receber atualizações de valores para uma determinada notícia.

A estrutura MqlCalendarValue disponibiliza método para obter e verificar valores a partir dos campos actual_value, forecast_value, prev_value e revised_prev_value. Se o valor do campo não estiver definido, ele armazenará o valor LONG_MIN (-9223372036854775808).

Deve-se ter em mente que os valores nestes campos são armazenados multiplicados por um fator de um milhão. Isso significa que quando as funções CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent e CalendarValueLast recebem valores em MqlCalendarValue, é necessário verificar se os valores LONG_MIN desses campos são iguais; e se no campo estiver definido o valor, para obter o valor deve-se dividir o valor do campo por 1000 000 (um milhão). Outra maneira de obter valores é verificar e obter valores por funções da própria estrutura MqlCalendarValue.

Um exemplo de um EA que ouve a publicação de eventos do Calendário Econômico:

```
#property description "Exemplo de uso da função CalendarValueLast"
#property description " para criar um ouvinte de evento do Calendário Econômico."
#property description "Para fazer isso, você precisa obter o ID atual da alteração"
#property description " base do Calendário. Em seguida, obter segundo este identificac
#property description " apenas novos eventos através de pesquisa no temporizador"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create timer
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- destroy timer
   EventKillTimer();
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- ID da alteração da base do Calendário
   static ulong calendar_change_id=0;
//--- sinal da primeira inicialização
   static bool first=true;
//--- array de valores do evento
   MqlCalendarValue values[];
//--- inicialize - obtenha o calendar_change_id atual
   if(first)
     {
      //--- obtenha o ID da alteração da base do Calendário
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- este bloco de código não pode ser executado na primeira inicialização,
         PrintFormat("%s: Obtido ID atual da base do calendário: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- defina o sinalizador e saia antes do próximo evento do temporizador
         first=false;
         return;
        }
      else
        {
         //--- dados não recebidos (para a primeira execução, isso é normal), verifiqu
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Obtido ID atual da base do calendário: change_id=%d",
                        __FUNCTION__,calendar_change_id);
            //--- defina o sinalizador e saia antes do próximo evento do temporizador
            first=false;
            //--- agora você tem o valor calendar_change_id
            return;
           }
         else
           {
            //--- isso é realmente um erro
```

```
            PrintFormat("%s: Falha ao obter eventos em CalendarValueLast. Código de er
                        __FUNCTION__,error_code);
            //--- falha ao trabalhar, tente inicializar novamente na próxima vez que c
            return;
           }
        }
     }

//--- você tem o último valor conhecido do ID de mudança do calendário (change_id)
   ulong old_change_id=calendar_change_id;
//--- verifique se apareceram novos eventos do Calendário
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Obtidos novos eventos do Calendário: %d",
                  __FUNCTION__,ArraySize(values));
      //--- exiba no Diário informações do array values
      ArrayPrint(values);
      //--- exiba no Diário os valores do ID do Calendário - o anterior e o novo
      PrintFormat("%s: Anterior change_id=%d, novo change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- exiba no Diário novos eventos
      ArrayPrint(values);
      /*
     escreva aqui o seu código que irá processar o surgimento dos eventos
      */
     }
//---
  }
/*
  Exemplo para a operação do "ouvinte":
  OnTimer: Obtido ID atual da base do calendário: change_id=33281792
  OnTimer: Obtidos novos eventos para o Calendário: 1
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Anterior change_id=33281792, novo change_id=33282048
        [id] [event_id]              [time]            [period] [revision] [actual_val
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -5077
  OnTimer: Obtidos novos eventos para o Calendário: 1
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -9223372036
  OnTimer: Anterior change_id=33282048, novo change_id=33282560
        [id] [event_id]              [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -9223372036

*/
```

Veja também

© 2000-2024, MetaQuotes Ltd.

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2024, MetaQuotes Ltd.

Acesso a séries de tempo e indicadores

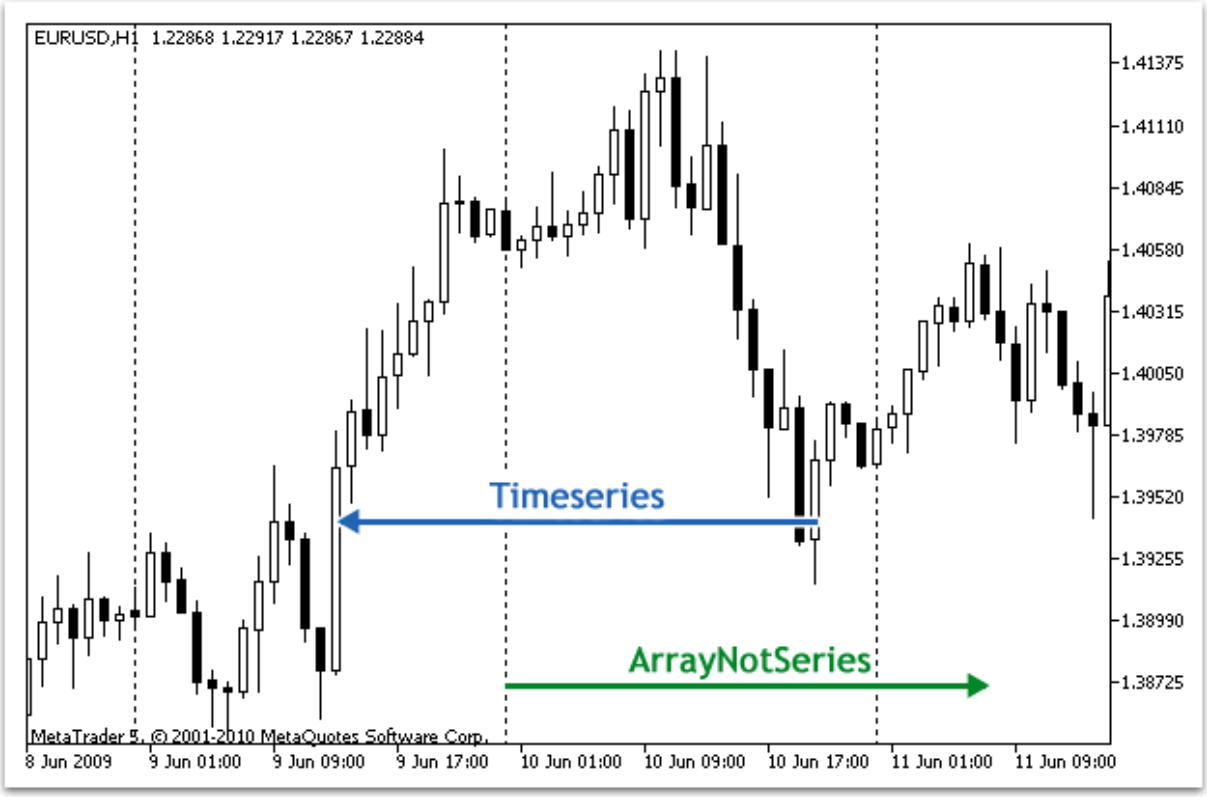
Estas são funções para trabalhar com séries de tempo e indicadores. Uma série de tempo difere de um array de dados usual pela sua ordenação reversa - elementos de séries de tempo são indexados do final de um array para seu início (a partir do mais recente para o mais antigo). Para copiar valores de séries de tempo e dados de indicadores, é recomendável usar somente arrays dinâmicos, porque as funções de cópia estão projetadas para alocar o tamanho necessário dos arrays que recebem valores.

Existe uma **importante exceção** a esta regra: se valores de séries de tempo e indicadores precisam ser copiados com freqüência, por exemplo, a cada chamada de OnTick() em Expert Advisors ou a cada chamada de OnCalculate() em indicadores, neste caso seria melhor usar arrays alocados estaticamente, porque **operações de alocação de memória** para arrays dinâmicos **requerem um tempo adicional**, e isso terá efeitos durante os processos de teste e otimização.

Ao usar funções de acesso a valores de séries de tempo e indicadores, a direção de indexação deve ser leva em consideração. Isso é descrito na seção Direção de indexação em arrays e séries de tempo.

O acesso a dados de indicadores e séries de tempo é implementado independentemente do fato dos dados solicitados estarem prontos ou não (o chamado acesso assíncrono). Isso é extremamente importante para os cálculos de um indicador customizado, porque se não houver dados, as funções de tipo *Copy...()* retornam imediatamente um erro. Contudo, ao acessar dados a partir de Expert Advisors e scripts, as várias tentativas para receber dados são feitos com uma pausa pequena, cujo objetivo é fornecer algum tempo necessário para baixar a série de tempo requerida para calcular valores de um indicador.

A seção Organizando Acesso a Dados descreve detalhes de recebimento, armazenamento e solicitação de dados de preço no terminal cliente MetaTrader 5.



© 2000-2024, MetaQuotes Ltd.

É historicamente aceito que um acesso a dados de preço em um array é realizado a partir do final dos dados. Fisicamente, os novos dados são sempre escritos no final do array, mas o índice do array é sempre igual a zero. O índice 0 no array de série de tempo denota dados da barra corrente, isto é, a barra que correspondente ao intervalo de tempo não terminado nesta janela de tempo.

Uma janela de tempo é o período de tempo, durante o qual uma única barra de preço é formada. Existem 21 janelas de tempo padrão predefinidas.

Função	Ação
SeriesInfoInteger	Retorna informação sobre o estado dos dados históricos
Bars	Retorna o número de barras do histórico para um ativo e período especificados
BarsCalculated	Retorna o número de dados calculados em um buffer de indicador ou -1 em caso de erro (os dados não foram calculados ainda)
IndicatorCreate	Retorna o handle para o indicador técnico especificado criado por um array de parâmetros de tipo MqlParam
IndicatorParameters	Baseado no handle especificado, retorna o número de parâmetros de entrada do indicador, bem como os valores e tipos dos parâmetros
IndicatorRelease	Remove um handle de indicador e libera o bloco de cálculo do indicador, se ele não for usado por ninguém mais
CopyBuffer	Copia dados de um buffer especificado de um indicador especificado em um array
CopyRates	Obtém dados históricos da estrutura Rates para um ativo e período especificados em um array
CopySeries	Obtém no conjunto de arrays especificado as séries temporais sincronizadas da estrutura MqlRates para o símbolo-período especificado na quantidade especificada.
CopyTime	Obtém dados históricos sobre hora de abertura de barras para um ativo e período especificados em um array
CopyOpen	Obtém dados históricos sobre preço de abertura de barras para um ativo e período especificados em um array
CopyHigh	Obtém histórico de dados sobre preço de barra máximo para um ativo e período especificados em um array
CopyLow	Obtém histórico de dados sobre preço de barra minimo para um ativo e período especificados em um array
CopyClose	Obtém histórico de dados sobre preço de fechamento de barra para um ativo e período especificados em um array
CopyTickVolume	Obtém histórico de dados sobre volumes de tick para um ativo e período especificados em um array
CopyRealVolume	Obtém dados históricos sobre volumes de negociação para um ativo e período especificados em um array

© 2000-2024, MetaQuotes Ltd.

Função	Ação
CopySpread	Obtém histórico de dados sobre spreads para um ativo e período especificados em um array
CopyTicks	Obtém, dentro da matriz ticks_array, ticks no formato MqlTick
CopyTicksRange	Obtém, na matriz ticks_array, ticks em formato MqlTick
iBars	Retorna o número de barras - no histórico - do símbolo e do período correspondentes
iBarShift	Retorna o índice da barra na qual o tempo especificado coincide
iClose	Retorna o valor do preço de fechamento da barra (indicado pelo parâmetro shift) do gráfico correspondente
iHigh	Retorna o valor do preço máximo da barra (especificado pelo parâmetro shift) do gráfico correspondente
iHighest	Retorna o índice do maior valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente
iLow	Retorna o valor do preço mínimo da barra (indicado pelo parâmetro shift) do gráfico correspondente
iLowest	Retorna o índice do menor valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente
iOpen	Retorna o valor do preço de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente
iTime	Retorna o valor do tempo de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente
iTickVolume	Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente
iRealVolume	Retorna o valor do volume real da barra (indicado pelo parâmetro shift) do gráfico correspondente
iVolume	Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente
iSpread	Retorna o valor do spread da barra (indicado pelo parâmetro shift) do gráfico correspondente

Apesar do fato de que pelo uso da função ArraySetAsSeries() é possível definir em arrays acesso aos elementos como em séries de tempo, deve ser lembrado que os elementos de arrays são fisicamente armazenados na mesma e única ordem - somente a direção de indexação muda. Para demonstrar este fato, vamos apresentar um exemplo:

```
   datetime TimeAsSeries[];
//--- define acesso ao array como uma série de tempo
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
```

© 2000-2024, MetaQuotes Ltd.

```
    if(copied<=0)
      {
       Print("A operação de cópia dos valores do tempo de abertura para as últimas 10 k
       return;
      }
    Print("TimeCurrent =",TimeCurrent());
    Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
    int size=ArraySize(TimeAsSeries);
    for(int i=0;i<size;i++)
      {
       Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
      }

    datetime ArrayNotSeries[];
    ArraySetAsSeries(ArrayNotSeries,false);
    ResetLastError();
    copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
    if(copied<=0)
      {
       Print("A operação de cópia dos valores do tempo de abertura para as últimas 10 k
       return;
      }
    size=ArraySize(ArrayNotSeries);
    for(int i=size-1;i>=0;i--)
      {
       Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
      }
```

Como resultado nós conseguimos a seguinte saída:

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
```

© 2000-2024, MetaQuotes Ltd.

```
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Como vemos a partir da saída, a medida que o índice do array TimeAsSeries aumenta, o valor da hora do índice diminui, isto é, nós nos movemos do presente para o passado. Para o array comum ArrayNotSeries o resultado é diferente - a medida que o índice cresce, nós nos movemos do passado para o presente.

Veja também

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2024, MetaQuotes Ltd.

Direção de Indexação em Arrays, Buffers e Séries Temporais

A indexação padrão de todos os arrays e buffers de indicador é da esquerda para direita. O índice do primeiro elemento é sempre igual a zero. Assim, o primeiro elemento de um array ou buffer de indicador com índice 0 está por padrão na posição mais a esquerda, enquanto o último elemento está na posição mais a direita.

Um buffer de indicador é um array dinâmico de tipo double, cujo tamanho é gerenciado pelos terminais clientes, de modo que ele sempre corresponde ao número de barras do indicador sobre o qual está calculando. Um array dinâmico usual de tipo double é atribuído como um buffer de indicador usando a função SetIndexBuffer(). Buffers de indicador não requerem que seus tamanhos sejam definidos usando a função ArrayResize() - isso será feito pelo sistema de execução do terminal.

Séries de tempo são array com indexação reversa, isto é, o primeiro elemento de uma série de tempo está na posição mais a direita, e o último elemento está na posição mais a esquerda. Em uma série de tempo sendo usada para armazenar dados de preços históricos e que contém informações de hora, os dados mais recentes são colocados na posição mais a direita da série de tempo, enquanto que os dados mais antigos na posição mais a esquerda.

Assim o elemento da série de tempo com índice 0 contém a informação sobre a última cotação de um ativo. Se uma série de tempo contém dados sobre uma janela de tempo diária, os dados correntes do dia ainda não terminado ficam localizados na posição zero, e na posição com índice 1 contém os dados de ontem.

Alterando a Direção de Indexação

A função ArraySetAsSeries() permite alterar o método de acessar elementos de um array dinâmico; no entanto, a ordem física de armazenamento de dados na memória do computador não é alterada. Esta função simplesmente altera o método de endereçar elementos de um array, assim quando copiar um array em um outro usando a função ArrayCopy(), o conteúdo do array recipiente não dependerá da direção de indexação no array fonte.

A direção de indexação não pode ser alterada para arrays alocados estaticamente. Mesmo se um array for passado como um parâmetro para uma função, tentar alterar a direção de indexação dentro desta função não produzirá nenhum efeito.

Para buffers de indicador, assim como para arrays comuns, a direção de indexação pode ser definida no sentido contrário (como em séries de tempo), isto é, uma referência à posição zero no buffer de indicador retornará o último valor no correspondente buffer de indicador e isso não corresponderá ao valor do indicador na barra mais recente. Contudo, a localização física das barras do indicador não será alterada.

Recebendo Dados de Preço em Indicadores

Cada indicador customizado deve necessariamente conter a função OnCalculate(), cujos dados de preço requeridos para calcular valores em buffers de indicadores são passados. A direção de indexação nestes arrays passados pode ser descoberta usando a função ArrayGetAsSeries().

Arrays passados para a função refletem dados de preço, isto é, estes arrays têm o sinal de uma série de preço e a função ArrayIsSeries() retornará true ao verificar estes arrays. Contudo, de qualquer forma a direção de indexação deve ser verificada somente pela função ArrayGetAsSeries().

© 2000-2024, MetaQuotes Ltd.

A fim de não ficar dependente de valores default, a função ArraySetAsSeries() deve ser incondicionalmente chamada para os arrays com os quais você vai trabalhar, e definir a direção requerida.

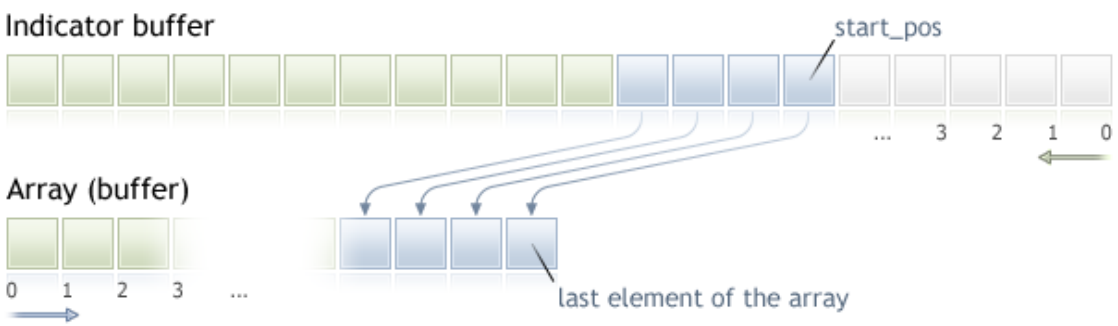
Recebendo Dados de Preço e Valores de Indicador

A direção de indexação padrão de todos os arrays em Expert Advisores, indicadores e scripts é da esquerda para direita. Se necessário, em qualquer programa mql5 você pode solicitar valores de séries de tempo de qualquer ativo e janela de tempo, bem como valores de indicadores calculados sobre qualquer ativo e janela de tempo.

Use as funções Copy...() para este propósito:

- CopyBuffer - copiar valores de um buffer de indicador para um array de tipo double;
- CopyRates - copiar histórico de preços para um array de estruturas MqlRates;
- CopyTime - copiar valores de Hora para um array de tipo datetime;
- CopyOpen - copiar valores de Abertura para um array de tipo double;
- CopyHigh - copiar valores de Máximo para um array de tipo double;
- CopyLow - copiar valores de Mínimo para um array de tipo double;
- CopyClose - copiar valores de Fechamento para um array de tipo double;
- CopyTickVolume - copiar volumes de tick para um array de tipo long;
- CopyRealVolume - copiar volumes de capital (equity) para um array de tipo long;
- CopySpread - copiar o histórico de spreads para um array de tipo int;

Todas estas funções trabalham de forma similar. Vamos considerar o mecanismo de obtenção de dados no exemplo de CopyBuffer(). É implícito que a direção de indexação dos dados requeridos é de série de tempo, e a posição com índice 0 (zero) armazena dados da barra corrente ainda incompleta. A fim de obter acesso a estes dados nós precisamos copiar o volume necessário de dados em um array recipiente, por exemplo, em um array de *buffer*.



Para copiar nós precisamos especificar a posição de início no array fonte, início a partir do qual dados serão copiados para o array recipiente. Em caso de sucesso, o número especificado de elementos serão copiados para o array recipiente do array fonte (do buffer de indicador neste caso). Independentemente do valor de indexação definido no array recipiente, a cópia é sempre realizada como exibida na figura abaixo.

© 2000-2024, MetaQuotes Ltd.

Se é esperado que um grande número de iterações seja tratado por um loop, é recomendável que se verifique a ocorrência de uma finalização de programa forçada usando a função IsStopped().

```
int copied=CopyBuffer(ma_handle,// manipulador do indicador
                      0,        // O índice do buffer do indicador
                      0,        // Posição início para copiar
                      number,   // Número de valores para copiar
                      Buffer    // O array que recebe os valores
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- Obtém o valor para o índice k
   double value=Buffer[k];
   // ...
   // trabalha com o valor
   k++;
  }
```

Exemplo:

```
input int per=10; // período do expoente
int ma_handle;    // manipulador do indicador
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// manipulador do indicador
                         0,        // índice do buffer do indicador
                         0,        // posição de ínicio da cópia
                         10,       // número de valores para copiar
                         ema       // array que recebe os valores
                         );
   if(copied<0) return;
// .... outro código
  }
```

© 2000-2024, MetaQuotes Ltd.

Também Veja

Organizando Acesso aos Dados

© 2000-2024, MetaQuotes Ltd.

Organizando Acesso aos Dados

Nesta seção, questões associados a obtenção, armazenamento e solicitação de dados de preços (séries de tempo) são consideradas.

Recebendo Dados de um Servidor de Negociação

Antes de dados de preços ficarem disponíveis no terminal MetraTrader 5, eles devem ser recebidos e processados. Para receber dados, uma conexão com o servidor de negociação MetaTrader 5 deve ser estabelecida. Dados são recebidos na forma de blocos empacotados de barras de um minuto do servidor sob a solicitação de um terminal.

O mecanismo de referência de servidor para solicitar dados não depende de como a solicitação foi iniciada - por um usuário ao navegar em um gráfico ou por meio de um programa na linguagem MQL5.

Armazenando Dados Intermediários

Dados recebidos de um servidor são automaticamente desempacotados e salvos no formato intermediário HCC. Os dados de cada ativo são escritos em uma pasta separada: *terminal_directory*\bases*server_name*\history*symbol_name*. Por exemplo, dados sobre EURUSD recebidos do servidor MetaQuotes-Demo serão armazenados em *terminal_directory*\bases\MetaQuotes-Demo\history\EURUSD\.

Os dados são escritos em arquivos com extensão .hcc. Cada arquivo armazena dados de barras de um minuto para um ano. Por exemplo, o arquivo nomeado 2009.hcc na pasta EURUSD contém barras de um minuto de EURUSD para o ano de 2009. Estes arquivos são usados para preparar dados de preço para todas as janelas de tempo e não são destinados para acesso direto.

Obtendo Dados em uma Janela de Tempo Necessária a partir dos Dados Intermediários

Os arquivo HCC intermediários são usados como fonte de dados para construir dados de preço para janelas de tempo solicitadas no formato HC. Dados de formato HC são séries de tempo que são maximamente preparados para acesso rápido. Eles são criados sob solicitação de um gráfico ou um programa MQL5. O volume de dados não deve exceder o valor do parâmetro "Máx. barras no gráfico". Os dados são armazenados para posterior uso em arquivos com extensão hc.

Para economizar recursos, os dados em uma janela de tempo são armazenados e guardamos em RAM somente se necessário. Se não for chamado por um longo tempo, eles são liberados da RAM e salvos em um arquivo. Para cada janela de tempo, os dados são preparados independentemente se existem dados prontos ou não para outras janelas de tempo. Regras para formação e acesso aos dados são as mesmas para todas as janelas de tempo. Isso significa que apesar da unidade de dados armazenada em HCC ser de um minuto, a disponibilidade de dados HCC não significa a disponibilidade de dados na janela de tempo M1 como HC no mesmo volume.

O recebimento de novos dados de um servidor chama automaticamente a atualização de dados de preço usados em formato HC de todas as janelas de tempo. Isso também leva ao recálculo de todos os indicadores que implicitamente usam estes dados como dados de entrada para cálculos.

Parâmetro "Máx. barras no gráfico"

© 2000-2024, MetaQuotes Ltd.

O parâmetro "Máx bars no gráfico" restringe o número de barras em formato HC disponível para gráficos, indicadores e programas mql5. Isso é válido para todas as janelas de tempo disponíveis e serve, primeiramente, para economizar recursos do computador.

Ao definir um grande valor para este parâmetro, deve ser lembrado que se dados de preço de histórico longo para pequenas janelas de tempo estiverem disponíveis, a memória usada para armazenadas a série de preços e buffers de indicadores podem se tornar centenas de megabytes e alcançar a restrição de RAM do programa terminal cliente (2Gb para aplicativos 32-bit do MS Windows).

A alteração do parâmetro "Máx. barras no gráfico" produz efeito após o terminal cliente ser reiniciado. A alteração deste parâmetro não causa referência automática a um servidor por dados adicionais, e nem formação de barras adicionais de séries de tempo. Dados de preço adicionais são solicitados ao servidor, e séries de preço são atualizadas levando em conta a nova limitação, em caso de rolagem de um gráfico para uma área sem dados, ou quando dados são solicitado por um programas MQL5 .

O volume de dados solicitados ao servidor corresponde ao número solicitado de barras de uma da janela de tempo com o parâmetro "Max. barras em gráfico" levado em consideração. A restrição definida por este parâmetro não é absoluta, e em alguns casos o número de barras disponíveis para uma janela de tempo pode ser um pouco maior que o valor corrente do parâmetro.

Disponibilidade de Dados

A presença de dados no formato HCC ou mesmo no formato preparado HC não significa sempre a absoluta disponibilidade destes dados para serem exibidos em um gráfico ou usados em um programa mql5.

Ao acessar dados de preços ou valores de indicadores a partir de um programa mql5 deve ser lembrado que a sua disponibilidade em um certo momento do tempo ou iniciando a partir de um certo momento de tempo não é garantida. Isso está relacionado com fato de que para economizar recursos, a cópia completa dos dados necessários para um programa mql5 não é armazenada no MetaTrader 5; apenas um acesso direto à base de dados do terminal é fornecida.

O histórico de preços para todas as janelas de tempo é construído a partir de dados comuns em formato HCC, e qualquer atualização dos dados a partir de um servidor conduz à atualização de dados para todas as janelas de tempo e ao recálculo dos indicadores. Devido a isso, o acesso aos dados pode ficar fechado, mesmo se estes dados estiverem disponíveis a um momento atrás.

Sincronização dos Dados do Terminal com os Dados do Servidor

Já que um programa mql5 pode chamar dados de qualquer ativo e janela de tempo, existe a possibilidade que os dados de uma série de tempo necessária não esteja formado ainda no terminal ou o preço necessário não esteja sincronizado com o servidor de negociação. Neste caso é difícil predizer o tempo de latência.

Algoritmos usando ciclos de latência não são a melhor solução. A única exceção neste caso são os scripts, porque eles não nenhuma escolha de algoritmo alternativo porque eles não tem manipuladores de evento. Para indicadores customizados tais algoritmos, bem como quaisquer outros ciclos de latência são fortemente não recomendados, porque levam a finalização do cálculo de todos os indicadores e qualquer outro manipulador de dados de preço do ativo.

Para Expert Advisor e indicadores, é melhor usar o modelo de eventos de manipulação. Se durante a manipulação dos eventos OnTick() ou OnCalculate(), o recebimento de dados para a janela de tempo

© 2000-2024, MetaQuotes Ltd.

requerida falhar, você deve sair do manipulador de evento, confiando na disponibilidade de acesso durante a próxima chamada do manipulador.

Exemplo de um Script para Adicionar Histórico

Vamos considerar um exemplo de um script que executa uma solicitação para receber histórico para o ativo selecionado de um servidor de negociação. O script é projetado para executar em um gráfico de um ativo selecionado; a janela de tempo não importa, porque como foi mencionado acima, dados de preço são recebidos de um servidor de negociação como pacotes de dados de um minuto, a partir dos quais qualquer série de tempo predefinida é construída.

Escreve todas as ações relacionadas a recepção de dados como uma função separada, CheckLoadHistory(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

A função CheckLoadHistory() é projetada como uma função universal que pode ser chamada de qualquer programa (Expert Advisor, script ou indicador); e portanto ela solicita três parâmetros de entrada: nome do ativo, período e data de inicio para indicar o começo do histórico de preço que você necessita.

Insira as verificações necessárias no código da função antes de solicitar o histórico faltante. Primeiramente, nós devemos assegurar que o nome do ativo e valor de período estão corretos:

```
   if(symbol==NULL || symbol==="") symbol=Symbol();
   if(period==PERIOD_CURRENT)      period=Period();
```

Então vamos nos assegurar que o ativo está disponível na janela Observador de Mercado, isto é, o histórico para o ativo estará disponível durante o envio de uma solicitação a um servidor de negociação. Se não houver tal ativo no Observador de Mercado, adicionar ele usando a função SymbolSelect().

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Agora nós devemos receber a data de início do histórico disponível para o par ativo/período indicado. Talvez, o valor do parâmetro de entrada startdate, passado para CheckLoadHistory(), já esteja disponível no histórico; então a solicitação a um servidor de negociação não é necessária. A fim de obter a primeira data para o ativo-período, a função SeriesInfoInteger() com o modificador SERIES_FIRSTDATE é usada.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

A próxima importante verificação é o tipo do programa, a partir do qual a função é chamada. Note que enviar uma solicitação de atualização de série de tempo com o mesmo período do indicador, que chama a atualização, não é desejável. O problema de solicitar dados sobre o mesmo ativo-período que o do indicador está relacionado ao fato de que a atualização de dados de histórico é realizada na

© 2000-2024, MetaQuotes Ltd.

mesma thread onde o indicador opera. Assim a possibilidade de ocorrência de conflito é alta. Para verificar isso use a função MQL5InfoInteger() com o modificador MQL5_PROGRAM_TYPE.

```
    if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
       return(-4);
```

Se todas as verificações tiverem passado com sucesso, faça a última tentativa de acesso sem se referir ao servidor de negociação Primeiramente, descubra a data de início, para qual dados de minuto no formato HCC estão disponíveis. Solicite este valor usando a função SeriesInfoInteger() com o modificador SERIES_TERMINAL_FIRSTDATE e compare novamente ele com o valor do parâmetro start_date.

```
    if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
      {
       //--- existe dados carregados para construir a série de tempo
       if(first_date>0)
         {
          //--- força a construção da série de tempo
          CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
          //--- verifica
          if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
             if(first_date>0 && first_date<=start_date) return(2);
         }
      }
```

Se após todas as verificações, a thread de execução estiver ainda no corpo da função CheckLoadHistory(), isso significa que existe uma necessidade de solicitar os dados de preço faltantes ao servidor de negociação. Primeiro, retorne o valor de "Máx. barras no gráfico" usando a função TerminalInfoInteger():

```
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Nós precisaremos dele para evitar solicitar dados extra. Então encontre a primeira data no sistema do ativo no servidor de negociação (independentemente do período) usando a já conhecida função SeriesInfoInteger() com o modificador SERIES_SERVER_FIRSTDATE.

```
    datetime first_server_date=0;
    while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
       Sleep(5);
```

Já que a solicitação é uma operação assíncrona, a função é chamada no ciclo com uma espera de 5 milissegundos até a variável first_server_date receber um valor, ou a execução do ciclo ser terminada por um usuário (IsStopped() retornará *true* neste caso). Vamos indicar um valor correto da data de início, começando a partir do qual nós solicitamos dados de preço de um servidor de negociação.

```
    if(first_server_date>start_date) start_date=first_server_date;
    if(first_date>0 && first_date<first_server_date)
       Print("Aviso: primeira data de servidor ",first_server_date," para",
symbol," não coincide com a primeira data de série ",first_date);
```

Se a data de início *first_server_date* do servidor for menor que o data de início *first_date* do ativo em formato HCC, a correspondente entrada será impressa no diário.

© 2000-2024, MetaQuotes Ltd.

Agora nós estamos prontos para fazer uma solicitação a um servidor de negociação por dados de preço faltantes. Faça a solicitação na forma de um ciclo e comece preenchendo seu corpo:

```
while(!IsStopped())
  {
   //1. espere pela sincronização entre a série de tempo reconstruída e o histórico
   //2. receba o número corrente de barra n desta série de tempo
   //    se bars for maior que Max_bars_in_char, nós podem sair, o trabalho está te
   //3. obter a data de início first_date na série de tempo reconstruída e comparar
   //    se first_date for menor que start_date, nós podemos sair, o trabalho está
   //4. solicitar a um servidor uma nova parte do histórico - 100 barras começando
  }
```

Os primeiros três pontos estão implementados pelos meios já conhecidos.

```
while(!IsStopped())
  {
   //--- 1. esperar até o processo de reconstrução da série de tempo acabar
   while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
      Sleep(5);
   //--- 2. solicitar quantas barras nós tempos
   int bars=Bars(symbol,period);
   if(bars>0)
     {
      //--- barras mais que podem ser desenhadas no gráfico, sair
      if(bars>=max_bars) return(-2);
      //--- 3. retorne a data de início corrente na série de tempo
      if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
         // data de início foi anterior a aquela solicitada, tarefa concluída
         if(first_date>0 && first_date<=start_date) return(0);
     }
   //4. Solicitar a um servidor uma nova parte do histórico - 100 barras começando
  }
```

Sobrou o quarto último ponto - solicitar histórico. Nós não podemos referenciar a um servidor diretamente, mas toda função-Copy inicia automaticamente uma solicitação de envio a um servidor, se o histórico em formato HCC não for suficiente. Já que a hora da primeira data de início na variável *first_date* é um simples e natural critério para avaliar o grau de execução de uma solicitação, então a forma mais fácil é usar a função CopyTime().

Ao chamar funções que copiam quaisquer dados de séries de tempo, deve ser notado que o parâmetro *start* (número da barra, começando a partir do qual dados de preço devem ser copiados) deve sempre estar dentro do histórico de terminal disponível. Se você tiver somente 100 barras, é inútil tentar copiar 300 barras começando da barra com o índice 500. Tal solicitação será entendida como um error e não será tratada, isto é, nenhum histórico adicional será carregado de um servidor de negociação.

Esta é a razão porque nós copiaremos 100 barras começando da barra com o índice *bars*. Isso fornecerá uma carga suave de histórico faltando doe um servidor de negociação. Na verdade, um pouco mais que as 100 barras solicitadas serão carregadas, já que o servidor envia um histórico superdimensionado.

```
int copied=CopyTime(symbol,period,bars,100,times);
```

© 2000-2024, MetaQuotes Ltd.

Após a operação de cópia, nós devemos analizar o número de elementos copiados. Se a tentativa falhar, então o valor *copiado* será igual a null e o valor do contador fail_cnt será aumentado em 1. Após 100 tentativas com falha, a operação da função será interrompida.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- verifica dados
      if(times[0]<=start_date)  return(0);  // o valor copiado é menor, pronto
      if(bars+copied>=max_bars) return(-2); // as barras são mais do que se pode deser
      fail_cnt=0;
     }
   else
     {
      //--- não mais que 100 tentativas falhando em sucessão
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

Então, não somente manipulação correta da situação corrente em cada momento de execução está implementada na função, mas também o código de finalização é retornado, que pode ser tratado depois chamando a função CheckLoadHistory() para obtenção de informações adicionais. Por exemplo, desta forma:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);                    break;
      case -2 : Print("More requested bars than can be drawn in the chart"); break;
      case -3 : Print("Execution stopped by user");                          break;
      case -4 : Print("Indicator mustn't load its own data");                break;
      case -5 : Print("Loading failed");                                     break;
      case  0 : Print("All data loaded");                                   break;
      case  1 : Print("Already available data in timeseries are enough");   break;
      case  2 : Print("Timeseries is built from available terminal data");  break;
      default : Print("Execution result undefined");
     }
```

O código completo da função pode ser encontrado no exemplo de um script que mostra a correta organização de acesso de quaisquer dados com a manipulação de resultados de solicitação.

Code:

```
//+------------------------------------------------------------------+
//|                                              TestLoadHistory.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- parâmetros de entrada
input string          InpLoadedSymbol="NZDUSD";   // Ativo a ser carregado
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Perídio a ser carregado
input datetime        InpStartDate=D'2006.01.01'; // Data de início
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Start load",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"from",InpSta
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Unknown symbol ",InpLoadedSymbol);             break;
      case -2 : Print("Requested bars more than max bars in chart"); break;
      case -3 : Print("Program was stopped");                        break;
      case -4 : Print("Indicator shouldn't load its own data");      break;
      case -5 : Print("Load failed");                                break;
      case  0 : Print("Loaded OK");                                  break;
      case  1 : Print("Loaded previously");                          break;
      case  2 : Print("Loaded previously and built");                break;
      default : Print("Unknown result");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("Primeira data ",first_date," - ",bars," barres");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- verifica ativo e período
   if(symbol==NULL || symbol==="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- verifica se o ativo está selecionado no Observador de Mercado
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
```

```
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- verifica se os dados estão presentes
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- não pede para carregar seus próprios dados se ele for um indicador
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym
      return(-4);
//--- segunda tentativa
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- existe dados carregados para construir a série de tempo
      if(first_date>0)
        {
         //--- força a construção da série de tempo
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- verifica
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- máximo de barras em um gráfico a partir de opções do terminal
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- carrega informações de histórico do ativo
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- corrige data de início para carga
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Aviso: primeira data de servidor ",first_server_date," para ",symbol,
            " não coincide com a primeira data de série ",first_date);
//--- carrega dados passo a passo
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- espera pela construção da série de tempo
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- pede por construir barras
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- pede pela primeira data
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- cópia da próxima parte força carga de dados
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- verifica dados
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- não mais que 100 tentativas com falha
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- interrompido
   return(-3);
  }
//+------------------------------------------------------------------+
//| Retorna valor de string do período                               |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
```

© 2000-2024, MetaQuotes Ltd.

```
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("período desconhecido");
  }
```

© 2000-2024, MetaQuotes Ltd.

SeriesInfoInteger

Retorna informações sobre o estado de dados históricos. Existem 2 variantes de chamada da função.

Retorna diretamente o valor da propriedade.

```
long  SeriesInfoInteger(
   string                      symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES             timeframe,        // período
   ENUM_SERIES_INFO_INTEGER    prop_id,          // identificador da propriedade
   );
```

Retorna true ou false dependendo do sucesso da execução da função.

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES             timeframe,        // período
   ENUM_SERIES_INFO_INTEGER    prop_id,          // ID da propriedade
   long&                       long_var          // variável a informação obtida
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

prop_id

[in] Identificador da propriedade solicitada, valor da enumeração ENUM_SERIES_INFO_INTEGER.

long_var

[out] Variável para onde o valor da propriedade solicitada é colocado.

Valor do Retorno

No primeiro caso, ela retorna valor do tipo long.

Para o segundo caso, ela retorna true, se a propriedade especificada estiver disponível e seu valor é colocada na variável long_var, caso contrário ela returna false. Para mais detalhes sobre um erro, chame a função GetLastError().

Exemplo:

```
void OnStart()
  {
//---
   Print("Total number of bars for the symbol-period at this moment = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("The first date for the symbol-period at this moment = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));
```

© 2000-2024, MetaQuotes Ltd.

```
  Print("The first date in the history for the symbol-period on the server = ",
        (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

  Print("Symbol data are synchronized = ",
        (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
}
```

© 2000-2024, MetaQuotes Ltd.

Bars

Retorna o número de barras no histórico para um ativo e período específicos. Existe 2 variantes de chamada da função.

Solicita todas as barras do histórico

```
int  Bars(
   string           symbol_name,     // nome do ativo
   ENUM_TIMEFRAMES  timeframe        // período
   );
```

Solicita as barras do histórico no intervalo de tempo selecionado

```
int  Bars(
   string           symbol_name,     // nome do ativo
   ENUM_TIMEFRAMES  timeframe,       // período
   datetime         start_time,      // data e hora de início
   datetime         stop_time        // data e hora de término
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_time

[in] Hora da barra correspondente ao primeiro elemento.

stop_time

[in] Hora da barra correspondente ao último elemento.

Valor do Retorno

Se os parâmetros start_time e stop_time estiverem definidos, a função retorna o número de barras no intervalo de tempo especificado, caso contrário retorna o número total de barras.

Observação

Se os dados para a série de tempo com parâmetros especificados não estiverem formados no terminal na hora que a função Bars() for chamada, ou dados da série de tempo não estiverem sincronizados com um servidor de negociação no momento da chamada da função, a função retorna o valor zero.

Ao solicitar o número de barras, no intervalo estabelecido de datas, são levados em conta apenas as barras cujo tempo de abertura está dentro desse intervalo. Por exemplo, se o dia da semana é sábado, ao pedir o número de barras semanais indicando start_time=última_terça_feira e stop_time=última_sexta_feira, a função retorna 0, pois, por um lado, o tempo de abertura no timeframe semanal sempre coincide com um domingo e, por outro, não há nenhuma barra semanal que fique dentro do intervalo estabelecido.

© 2000-2024, MetaQuotes Ltd.

Exemplo de solicitação do número de todas as barras no histórico:

```
int bars=Bars(_Symbol,_Period);
if(bars>0)
  {
   Print("Number of bars in the terminal history for the symbol-period at the momer
  }
else  //sem barras disponíveis
  {
   //--- dados sobre o ativo podem não estar sincronizados com os dados no servido
   bool synchronized=false;
   //--- contador de loop
   int attempts=0;
   // faz 5 tentativas de espera por sincronização
   while(attempts<5)
     {
      if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
         //--- sincronização feita, sair
         synchronized=true;
         break;
        }
      //--- aumentar o contador
      attempts++;
      //--- espera 10 milissegundos até a próxima iteração
      Sleep(10);
     }
   //--- sair do loop após sincronização
   if(synchronized)
     {
      Print("Number of bars in the terminal history for the symbol-period at the mo
      Print("The first date in the terminal history for the symbol-period at the mo
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
      Print("The first date in the history for the symbol on the server = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
     }
   //--- sincronização dos dados não aconteceu
   else
     {
      Print("Failed to get number of bars for ",_Symbol);
     }
  }
```

Exemplo de solicitação do número de barras no intervalo estabelecido:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // sexta-feira
datetime date2 = D'2016.09.05 00:00'; // segunda-feira
datetime date3 = D'2016.09.08 00:00'; // quinta-feira
//---
```

© 2000-2024, MetaQuotes Ltd.

```
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
Print("Número de barras: ",n); // Mostra "Número de barras: 8", na contagem levará
n=Bars(_Symbol,PERIOD_D1,date1,date2);
Print("Número de barras: ",n); // Mostra "Número de barras: 1", pois o tempo de abe
n=Bars(_Symbol,PERIOD_W1,date2,date3);
Print("Número de barras: ",n); // Mostra "Número de barras: 0", pois o tempo de abe
```

Veja Também

Funções de Processamento de Eventos

© 2000-2024, MetaQuotes Ltd.

BarsCalculated

Retorna o número de dados calculados para o indicador especificado.

```
int  BarsCalculated(
   int       indicator_handle,      // handle do indicador
   );
```

Parâmetros

indicator_handle

[in] O handle do indicador, retornado pela função do indicador correspondente.

Valor do Retorno

Retorna a quantidade de dados calculados no buffer de indicador, ou -1 em caso de erro (dados ainda não calculados)

Observação

A função é útil quando é necessário obter os dados do indicador imediatamente após sua criação (o handle do indicador está disponível).

Exemplo:

```
void OnStart()
  {
   double Ups[];
//--- define a ordenação de série de tempo para os arrays
   ArraySetAsSeries(Ups,true);
//--- cria handle para o indicador Fractal
   int FractalsHandle=iFractals(NULL,0);
//--- redefine o código de erro
   ResetLastError();
//--- tenta copiar os valores do indicador
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("Failed to copy upper fractals. Erro = ",GetLastError(),
         "i = ",i,"    copiado = ",copied);
         return;
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      else
        Print("Upper fractals copied",
        "i = ",i,"    copiado = ",copied);
    }
  else Print("Upper fractals copied. ArraySize = ",ArraySize(Ups));
}
```

© 2000-2024, MetaQuotes Ltd.

IndicatorCreate

A função retorna o handle de um indicador técnico especificado criado baseado no array de parâmetros de tipo MqlParam.

```
int  IndicatorCreate(
   string           symbol,                          // nome do ativo
   ENUM_TIMEFRAMES  period,                          // janela de tempo
   ENUM_INDICATOR   indicator_type,                  // tipo de indicador a partir d
   int              parameters_cnt=0,                // número de parâmetros
   const MqlParam&  parameters_array[]=NULL,         // array de parâmetros
   );
```

Parâmetros

symbol

[in] Nome de um ativo, sobre cujos dados o indicador é calculado. NULL significa o ativo corrente.

period

[in] O valor da janela de tempo pode ser um dos valores da enumeração ENUM_TIMEFRAMES, 0 significa a janela de tempo corrente.

indicator_type

[in] Tipo de indicador, pode ser um dos valores da enumeração ENUM_INDICATOR.

parameters_cnt

[in] O número de parâmetros passados no array parameters_array[]. Os elementos do array têm uma estrutura especial de tipo MqlParam. Por default, zero - parâmetros são passados. Se você especificar um número não zero de parâmetros, o parâmetro *parameters_array* é obrigatório. Você não pode passar mais que 64 parâmetros.

parameters_array[]=NULL

[in] Um array do tipo MqlParam, cujos elementos contém o tipo e valor de cada parâmetro de input de um indicador técnico.

Valor de retorno

Retorna o handle de um indicador técnico especificado, em caso de falha retorna INVALID_HANDLE.

Observação

Se o handle de indicador de tipo IND_CUSTOM for criado, o campo *type* do primeiro elemento do array de parâmetros de entrada *parameters_array* deve ter o valor TYPE_STRING da enumeração ENUM_DATATYPE, e o campo *string_value* do primeiro elemento deve conter o nome do indicador customizado. O indicador customizado deve estar compilado (arquivo com extensão EX5) e localizado no diretório MQL5/Indicators do terminal cliente ou em um subdiretório.

Indicadores que requerem teste são definidos automaticamente a partir da chamada da função iCustom(), se o correspondente parâmetro for definido através de um string constante. Para todos os outros casos (uso da função IndicatorCreate() ou uso de uma string não-constante no parâmetro de define o nome do indicador) a propriedade #property tester_indicator é requerida:

```
#property tester_indicator "indicator_name.ex5"
```

© 2000-2024, MetaQuotes Ltd.

Se primeira forma da chamada for usada em um indicador customizado, você pode adicionalmente indicar como último parâmetro sobre quais dados o indicador será calculado, ao passar os parâmetros de input. Se o parâmetro "Apply to" não for especificado explicitamente, o cálculo default é baseado no valores de PRICE_CLOSE.

Exemplo:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- criar iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- define ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- define ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- define ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- define applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- criar MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- criar iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- define ma_period rápido
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- define ma_period lento
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- define período de suavização por diferença
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- define o handle do indicador como applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- criar MACD baseado em média móvel
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- usa os indicadores
//--- . . .
//--- libera os indicatores (primeiro h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2024, MetaQuotes Ltd.

IndicatorParameters

Baseado no handle especificado, retorna o número de parâmetros de entrada do indicador, bem como os valores e tipos dos parâmetros.

```
int  IndicatorParameters(
   int               indicator_handle,      // handle do indicador
   ENUM_INDICATOR&   indicator_type,        // uma variável para receber o tipo do indi
   MqlParam&         parameters[]           // um array para receber parâmetros
   );
```

Parâmetros

indicator_handle

[in] O handle do indicador, para o qual você precisa saber o número de parâmetros sobre os quais o indicador é calculado.

indicator_type

[out] Uma variável de tipo ENUM_INDICATOR, na qual o tipo do indicador será escrito.

parameters[]

[out] Um array dinâmico para receber valores do tipo MqlParam, na qual a lista de parâmetros do indicador será escrito. O tamanho do array é retornado pela função IndicatorParameters().

Valor do Retorno

O número de parâmetros de entrada do indicador com o handle especificado. No caso de um erro, retorna -1. Para mais detalhes sobre o erro chame a função GetLastError().

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- O número de janelas no gráfico (ao menos uma janela principal está sempre preser
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- Varre as janelas do gráfico
   for(int w=0;w<windows;w++)
     {
      //--- O número de indicadores nesta janela/sub-janela
      int total=ChartIndicatorsTotal(0,w);
      //--- Pega todos os indicadores na janela
      for(int i=0;i<total;i++)
        {
         //--- Obtém o nome abreviado do indicador
         string name=ChartIndicatorName(0,w,i);
         //--- Obtém o handle do indicador
         int handle=ChartIndicatorGet(0,w,name);
         //--- Adicionar o log
```

© 2000-2024, MetaQuotes Ltd.

```
          PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
          //---
          MqlParam parameters[];
          ENUM_INDICATOR indicator_type;
          int params=IndicatorParameters(handle,indicator_type,parameters);
          //--- O cabeçalho da mensagem
          string par_info="Short name "+name+", type "
                          +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
          //---
          for(int p=0;p<params;p++)
            {
             par_info+=StringFormat("parâmetro %d: type=%s, long_value=%d, double_value
                                    p,
                                    EnumToString((ENUM_DATATYPE)parameters[p].type),
                                    parameters[p].integer_value,
                                    parameters[p].double_value,
                                    parameters[p].string_value
                                    );
            }
          Print(par_info);
         }
       //--- Pronto para todos os indicadores da janela
      }
//---
  }
```

Também Veja

ChartIndicatorGet()

© 2000-2024, MetaQuotes Ltd.

IndicatorRelease

A função remove um handle de indicador e libera o bloco de cálculo do indicador, se ele não for usado por ninguém mais.

```
bool  IndicatorRelease(
   int       indicator_handle,     // handle do indicador
   );
```

Valor de retorno

Retorna true no caso de sucesso, caso contrário retorna false.

Observação

A função permite remover um handle de indicador, se ele não mais for necessário, deste modo, economizando memória . O handle é removido imediatamente, o bloco de cálculo é excluído em algum momento (se ele não for chamado por ninguém mais).

Durante o funcionamento do testador de estratégias, a função IndicatorRelease() não é executada.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                      Test_IndicatorRelease.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parâmetros de entrada
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- armazenará o handle de indicador
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- cria um handle de indicador
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- exclui variável global
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

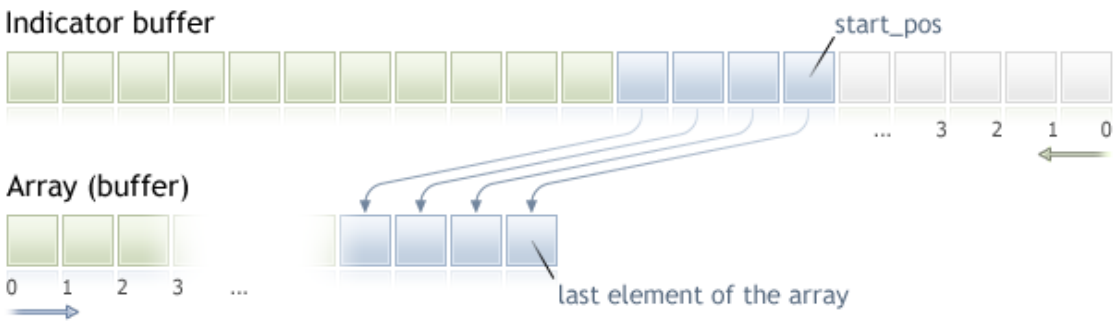
© 2000-2024, MetaQuotes Ltd.

```
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- se o valor da variável global não existir
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- obtém o valor do indicador nas últimas duas barras
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- array dinâmico para valores do indicador
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- lembra na variável global o valor da penúltima barra
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- libera o handle do indicador
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() falhou. Error ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet falhou. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2024, MetaQuotes Ltd.

CopyBuffer

Obtém dados de um buffer especificado de um certo indicador na quantidade necessária.



A contagem de elementos de dados copiados (buffer de indicador com o índice buffer_num) da posição de início é realizada do presente para o passado, isto é, posição de início 0 significa a barra corrente (valor do indicador para a barra corrente).

Ao copiar uma quantidade ainda desconhecida de dados, é recomendado usar um array dinâmico como um buffer[] recipiente, porque a função CopyBuffer() tenta alocar o tamanho do array receptor ao tamanho dos dados copiados. Se um buffer de indicador (array que é pré-alocado para armazenar valores de um indicador através da função SetIndexBufer()) for usado como um array recipiente buffer[], uma copia parcial é permitida. Um exemplo pode ser encontrado no indicador customizado Awesome_Oscillator.mql5 no pacote padrão do terminal.

Se você precisar fazer uma cópia parcial dos valores de um indicador em um outro array (um não buffer de indicador), você deve usar um array intermediário, para o qual o número desejado é copiado. Após isso, conduza uma cópia elemento a elemento do requerido número de valores para os lugares requeridos em um array de recepção a partir deste array intermediário.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyBuffer(
   int       indicator_handle,     // handle do indicador
   int       buffer_num,           // número do buffer do indicador
   int       start_pos,            // posição de início
   int       count,                // quantidade para copiar
   double    buffer[]              // array destino para copiar
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyBuffer(
   int       indicator_handle,     // handle do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
   int       buffer_num,          // número do buffer do indicador
   datetime  start_time,          // data e hora de início
   int       count,               // quantidade para copiar
   double    buffer[]             // array destino para copiar
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyBuffer(
   int       indicator_handle,    // handle do indicador
   int       buffer_num,          // número do buffer do indicador
   datetime  start_time,          // data e hora de início
   datetime  stop_time,           // data e hora de término
   double    buffer[]             // array destino para copiar
   );
```

Parâmetros

indicator_handle

[in] O handle do indicador, retornado pela função do indicador correspondente.

buffer_num

[in] O número do buffer do indicador.

start_pos

[in] A posição do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] Hora da barra, correspondente ao primeiro elemento.

stop_time

[in] Hora da barra, correspondente ao último elemento.

buffer[]

[out] Array de tipo double.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite.

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer3.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- buffers do indicador
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parameter AsSeries = ",AsSeries);
   Print("Buffer do indicador após SetIndexBuffer() é timeseries = ",
         ArrayGetAsSeries(MABuffer));
//--- define o nome abreviado do indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- define AsSeries(depende do parâmetro de entrada)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Buffer do indicador após ArraySetAsSeries(MABuffer,true); é timeseries = ",
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- verifica se todos os dados estão calculados
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- nós não podemos copiar todos os dados
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- o último valor é sempre copiado
      to_copy++;
     }
//--- tenta copiar
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

O exemplo acima ilustra como um buffer de indicador é preenchido com os valores de um outro buffer de indicador a partir de um indicador sobre o mesmo ativo/período.

Veja uma exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

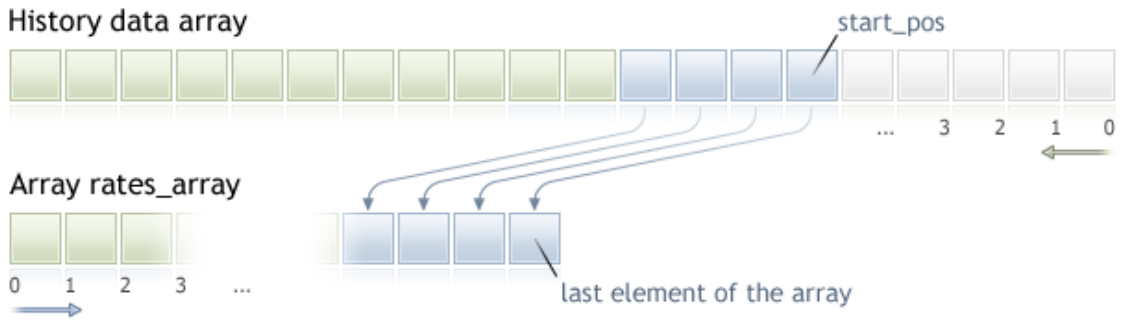
Também Veja

Propriedades de Indicadores Customizados, SetIndexBuffer

© 2000-2024, MetaQuotes Ltd.

CopyRates

Obtém dados históricos de estrutura MqlRates de um ativo-período especificado na quantidade especificada no array rates_array. A ordenação dos elementos dos dados copiados é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyRates(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posição de início
   int              count,             // quantidade de dados para copiar
   MqlRates         rates_array[]      // array destino para copiar
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyRates(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   int              count,             // quantidade de dados para copiar
   MqlRates         rates_array[]      // array destino para copiar
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyRates(
```

© 2000-2024, MetaQuotes Ltd.

```
   string          symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES timeframe,         // período
   datetime        start_time,        // data e hora de início
   datetime        stop_time,         // data e hora de término
   MqlRates        rates_array[]      // array destino para copiar
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_time

[in] Hora da barra do primeiro elemento para copiar.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

rates_array[]

[out] Array de tipo MqlRates.

Valor do Retorno

Retorna o número de elementos copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

Exemplo:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Barres copiados: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("Falha ao receber dados históricos para o símbolo ",Symbol());
  }
```

Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

© 2000-2024, MetaQuotes Ltd.

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

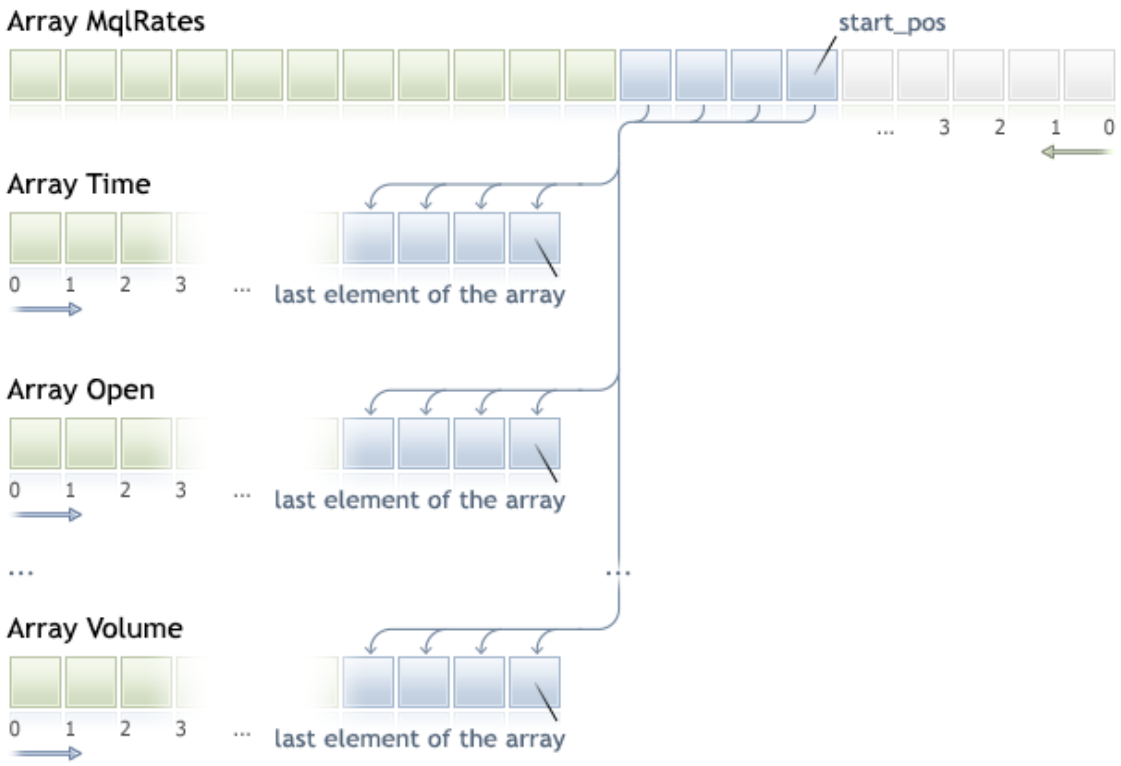
Também Veja

Estruturas e Classes, TimeToString, StringFormat

© 2000-2024, MetaQuotes Ltd.

CopySeries

Obtém no conjunto de arrays especificado as séries temporais sincronizadas da estrutura MqlRates para o símbolo-período especificado na quantidade especificada. Os itens da posição inicial são contados do presente para o passado, ou seja, uma posição inicial de 0 significa a barra atual.



Ao copiar antecipadamente uma quantidade desconhecida de dados, é recomendável usar como arrays receptores uma matriz dinâmica, já que se houver mais dados do que o array pode conter, é feita uma tentativa de redistribuir a matriz para que os dados solicitados se ajustem a todo o array.

Se uma quantidade pré-determinada de dados precisa ser copiada, é melhor fazê-lo em um buffer alocado estaticamente, para evitar a alocação global desnecessária de memória.

Não importa que propriedade o array receptor tenha, as_series=true ou as_series=false, os dados serão copiados para que o elemento da série temporal mais antiga seja copiado para o início da memória física alocada ao array.

```
int  CopySeries(
   string           symbol_name,        // nome do símbolo
   ENUM_TIMEFRAMES  timeframe,          // período
   int              start_pos,          // onde começamos
   int              count,              // quanto copiamos
   ulong            rates_mask,         // combinação de sinalizadores para indicar a s
   void&            array1[],           // matriz onde serão copiados os dados da prime
   void&            array2[]            // matriz onde os dados da segunda série tempo
   ...
   );
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

symbol_name

[in] Símbolo.

timeframe

[in] Período.

start_pos

[in] Número do primeiro item copiado.

count

[in] Número de itens a serem copiados.

rates_mask

[in] Combinação de sinalizadores a partir da enumeração ENUM_COPY_RATES .

array1, array2,...

[out] Matriz do tipo apropriado para recuperar as séries temporais a partir da estrutura MqlRates. A ordem dos arrays passados para a função deve corresponder à ordem dos campos na estrutura MqlRates.

Valor retornado

Número de elementos do array copiados ou -1 no caso de erro.

Observação

Se o intervalo de tempo solicitado estiver totalmente fora dos dados disponíveis no servidor, a função retorna -1. Se os dados solicitados estiverem fora do escopo TERMINAL_MAXBARS (número máximo de barras no gráfico), a função também retornará -1.

Ao solicitar dados do indicador, caso as séries temporais solicitadas ainda não tenham sido construídas ou precisem ser baixadas do servidor, a função retornará imediatamente -1, mas o próprio processo de carregamento/construção será iniciado.

Ao solicitar dados de um EA ou script, será iniciada um carregamento desde o servidor se o terminal não tiver esses dados localmente, ou será iniciada a construção da série temporal necessária se os dados puderem ser construídos a partir do histórico local, mas ainda não estiverem prontos. A função retornará a quantidade de dados que estarão prontos quando o tempo expirar, mas o carregamento do histórico continuará, e na próxima vez que uma solicitação semelhante for feita a função retornará mais dados.

Diferença entre CopySeries e CopyRates

A função CopySeries permite obter apenas as séries temporais necessárias de cada vez em diferentes arrays especificados, enquanto todos estão sincronizados entre si. Isso significa que todos os valores nos arrays resultantes em um índice específico N pertencerão à mesma barra no par Symbol/Timeframe especificado. Nesse caso, o programador não precisa garantir que todas as séries temporais recebidas sejam sincronizadas pelo tempo de abertura da barra.

© 2000-2024, MetaQuotes Ltd.

Ao contrário de CopyRates, que retorna o conjunto completo de séries temporais como um array MqlRates, a função CopySeries permite que o programador obtenha apenas as séries temporais necessárias como arrays separados. Para fazer isso, você precisa especificar uma combinação de sinalizadores para especificar o tipo de série temporal solicitado . A ordem dos arrays passados para a função deve corresponder à ordem dos campos na estrutura MqlRates:

```
struct MqlRates
  {
   datetime time;         // início de período
   double   open;         // preço de abertura
   double   high;         // preço mais alto do período
   double   low;          // preço mais baixo do período
   double   close;        // preço de fechamento
   long     tick_volume;  // volume de tick
   int      spread;       // spread
   long     real_volume;  // volume de negociação
  }
```

Assim, se for necessário obter os valores das séries temporais *time*, *close* e *real_volume* para as últimas 100 barras do Symbol/Timeframe atual, a chamada deve ser a seguinte:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

A ordem das matrizes é importante "*time*, *close*, *volume*" — deve estar de acordo com a ordem dos campos na estrutura MqlRates. Já a ordem dos valores na máscara *rates_mask* não importa, a máscara poderia ser assim:

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Exemplo:

```
//--- input parameters
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- arrays para obter séries temporais da estrutura de preços MqlRates
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- solicitamos preços de fechamento para um array do tipo double
   ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries  returns %d values. Error code=%d", res1, GetLastError()
   ArrayPrint(close);

//--- agora solicitamos mais preços de abertura e preços de fechamento em um array do
   ResetLastError();
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries  returns %d values. Error code=%d", res2, GetLastError()
   ArrayPrint(closef);
//--- comparamos os dados
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open       | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- Resultado
      1. CopySeries  returns 20 values. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.
      2. CopySeries  returns 20 values. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.
        | Time              |    Open       | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

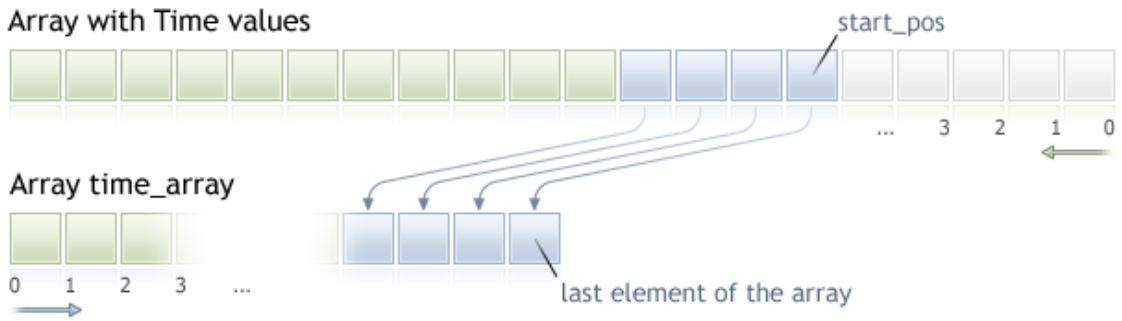
Ver também

Estruturas e classes, CopyRates

© 2000-2024, MetaQuotes Ltd.

CopyTime

A função obtém em time_array o histórico de dados da hora de abertura de barras para o par ativo-período especificado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyTime(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posição de início
   int               count,            // quantidade de dados para copiar
   datetime          time_array[]      // array destino para copiar as horas de abertura
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyTime(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // data e hora de início
   int               count,            // quantidade de dados para copiar
   datetime          time_array[]      // array destino para copiar horas de abertura
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyTime(
```

© 2000-2024, MetaQuotes Ltd.

```
    string          symbol_name,      // nome do ativo
    ENUM_TIMEFRAMES timeframe,        // período
    datetime        start_time,       // data e hora de início
    datetime        stop_time,        // data e hora de parada
    datetime        time_array[]      // array destino para copiar as horas de abertura
    );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

time_array[]

[out] Array de tipo datetime.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

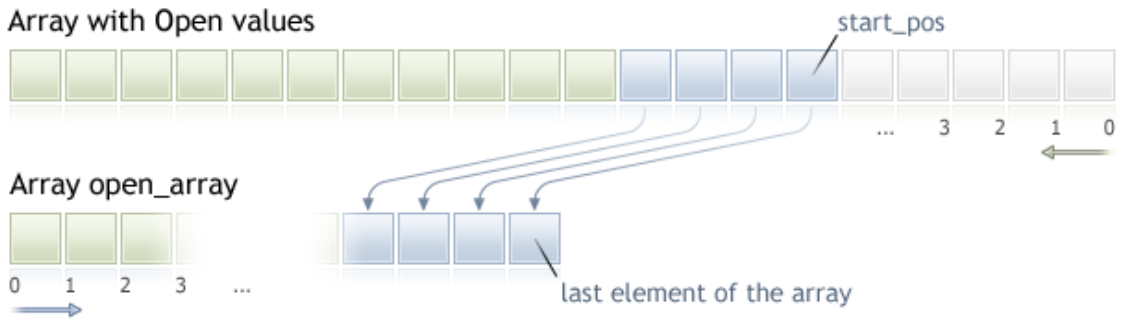
Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyOpen

A função obtém em open_array os dados históricos de preços de abertura de barras para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyOpen(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posição de início
   int               count,            // quantidade de dados para copiar
   double            open_array[]      // array destino para copiar preços de abertura
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyOpen(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // data e hora de início
   int               count,            // quantidade de dados para copiar
   double            open_array[]      // array destino para preços de abertura de barra
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyOpen(
```

© 2000-2024, MetaQuotes Ltd.

```
    string          symbol_name,      // nome do ativo
    ENUM_TIMEFRAMES timeframe,        // período
    datetime        start_time,       // data e hora de início
    datetime        stop_time,        // data e hora de parada
    double          open_array[]      // array destino para valores de abertura de bar
    );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] A hora de início para o último elemento para copiar.

open_array[]

[out] Array de tipo double.

Valor do Retorno

Retorna o número de elementos no array ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

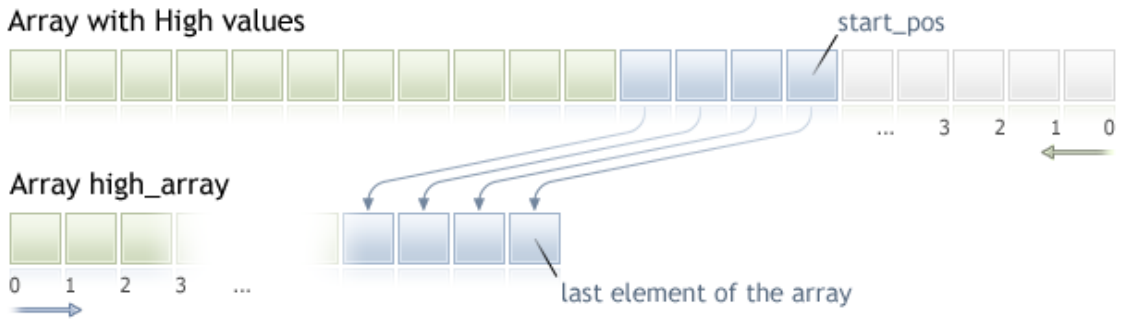
Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyHigh

A função obtém em high_array os dados históricos dos preços de barra mais altos para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyHigh(
   string            symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES   timeframe,         // período
   int               start_pos,         // posição de início
   int               count,             // quantidade de dados para copiar
   double            high_array[]       // array destino para copiar
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyHigh(
   string            symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES   timeframe,         // período
   datetime          start_time,        // data e hora de início
   int               count,             // quantidade de dados para copiar
   double            high_array[]       // array destino para copiar
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyHigh(
```

© 2000-2024, MetaQuotes Ltd.

```
    string          symbol_name,      // nome do ativo
    ENUM_TIMEFRAMES timeframe,        // período
    datetime        start_time,       // data e hora de início
    datetime        stop_time,        // data e hora de parada
    double          high_array[]      // array destino para copiar
    );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

high_array[]

[out] Array de tipo double.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

Exemplo:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "An example for output of the High[i] and Low[i]"
#property description "for a random chosen bars"

double High[],Low[];
//+------------------------------------------------------------------+
//| Obtém Mínimo para o índice de barra especificado                 |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//| Obtém o Máximo para o índice de barra especificado               |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- em cada tick nós imprimimos os valores de Máximo e Mínimo para a barra com índic
//--- que seja igual ao segundo, no qual o tick chegou
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

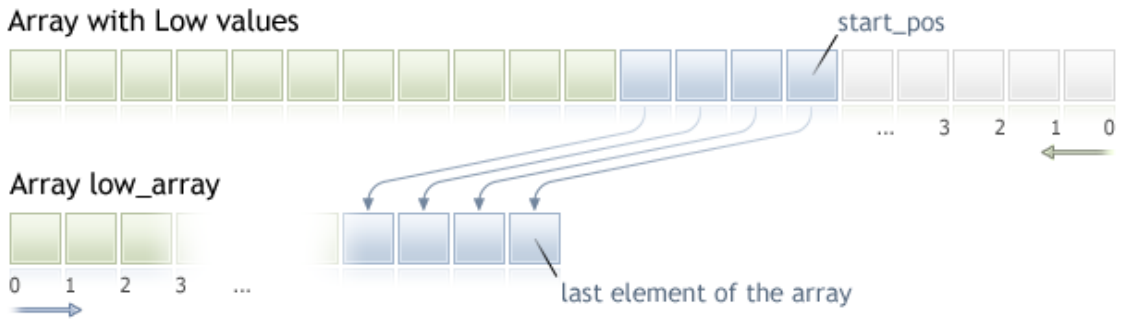
Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyLow

A função obtém em low_array os dados históricos de preços de barra mínimos para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyLow(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   int               start_pos,        // posição de início
   int               count,            // quantidade de dados para copiar
   double            low_array[]       // array destino para copiar
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyLow(
   string            symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES   timeframe,        // período
   datetime          start_time,       // data e hora de início
   int               count,            // quantidade de dados para copiar
   double            low_array[]       // array destino para copiar
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyLow(
```

© 2000-2024, MetaQuotes Ltd.

```
    string          symbol_name,      // nome do ativo
    ENUM_TIMEFRAMES timeframe,        // período
    datetime        start_time,       // data e hora de início
    datetime        stop_time,        // data e hora de parada
    double          low_array[]       // array destino para copiar
    );
```

Parâmetros

symbol_name

[in] Ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] Hora da barra, correspondente ao último elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

low_array[]

[out] Array de tipo double.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

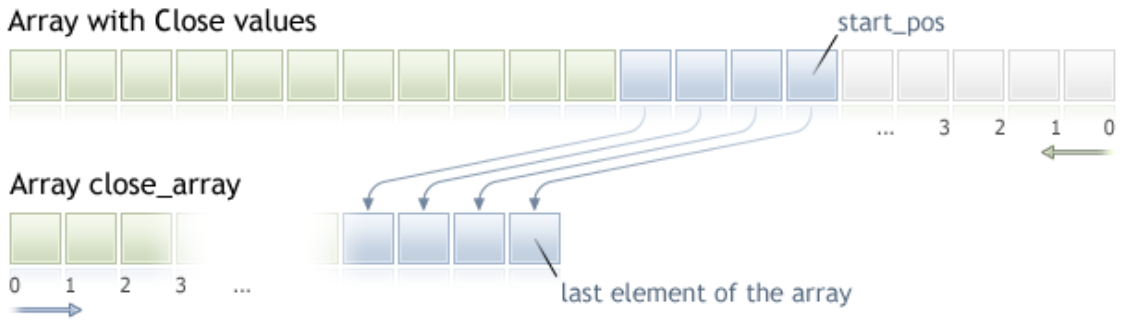
Também Veja

CopyHigh

© 2000-2024, MetaQuotes Ltd.

CopyClose

A função obtém em close_array os dados históricos de preços de fechamento de barra para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyClose(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posição de início
   int              count,             // quantidade de dados para copiar
   double           close_array[]      // array destino para copiar
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyClose(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   int              count,             // quantidade de dados para copiar
   double           close_array[]      // array destino para copiar
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyClose(
```

© 2000-2024, MetaQuotes Ltd.

```
    string          symbol_name,       // nome do ativo
    ENUM_TIMEFRAMES timeframe,         // período
    datetime        start_time,        // data e hora de início
    datetime        stop_time,         // data e hora de parada
    double          close_array[]      // array destino para copiar
    );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

close_array[]

[out] Array de tipo double.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

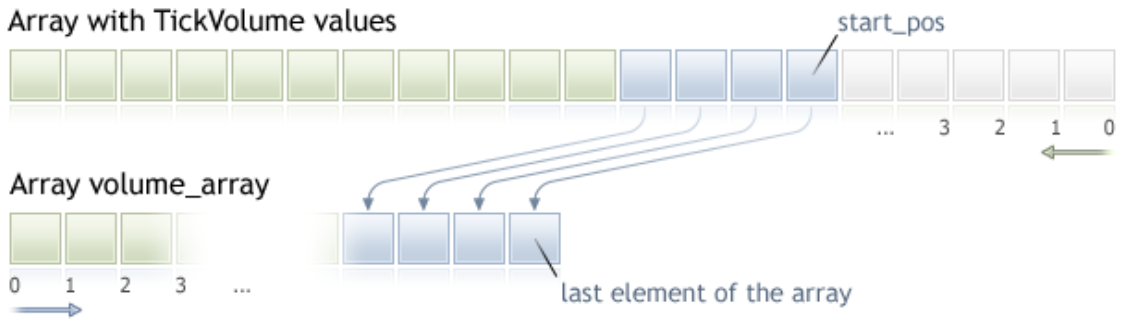
Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyTickVolume

A função obtém em volume_array os dados históricos de volumes de tick para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyTickVolume(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posição de início
   int              count,             // quantidade de dados para copiar
   long             volume_array[]     // array destino para volumes de tick
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyTickVolume(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   int              count,             // quantidade de dados para copiar
   long             volume_array[]     // array destino para volumes de tick
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyTickVolume(
```

© 2000-2024, MetaQuotes Ltd.

```
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   datetime         stop_time,         // data e hora de parada
   long             volume_array[]     // array destino para volumes de tick
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

volume_array[]

[out] Array de tipo long.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

Exemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      bars=3000;
//--- buffers do indicador
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2024, MetaQuotes Ltd.

```
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("We have received the following number of TickVolume values: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

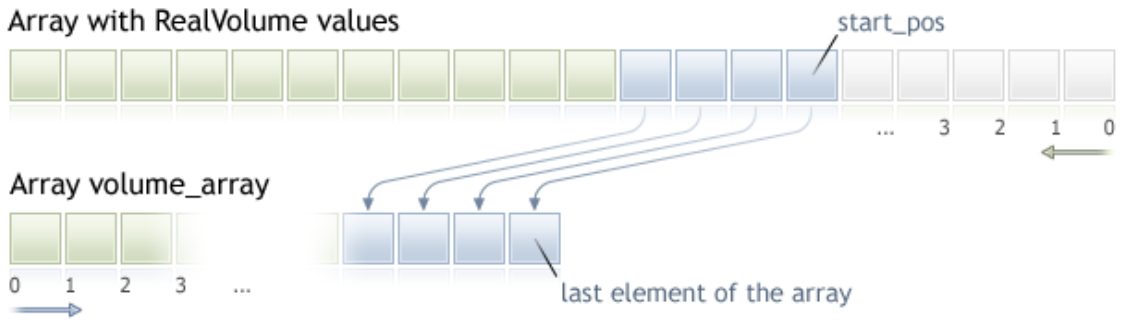
Veja um exemplo detalhado de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyRealVolume

A função obtém em volume_array os dados históricos de volumes de negociação para o par ativo-período selecionado na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopyRealVolume(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   int              start_pos,         // posição de início
   int              count,             // quantidade de dados para copiar
   long             volume_array[]     // array destino para valores de volume
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopyRealVolume(
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   int              count,             // quantidade de dados para copiar
   long             volume_array[]     // array destino para valores de volume
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopyRealVolume(
```

© 2000-2024, MetaQuotes Ltd.

```
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   datetime         stop_time,         // data e hora de parada
   long             volume_array[]     // array destino para valores de volume
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

volume_array[]

[out] Array de tipo long.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

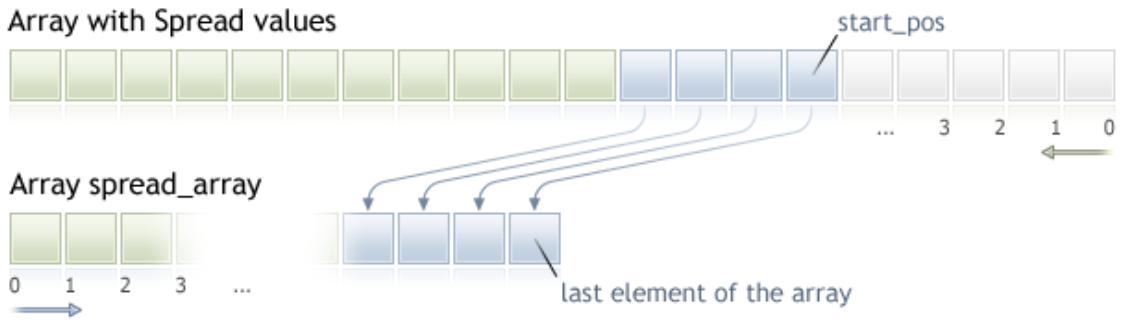
Veja um exemplo de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopySpread

A função obtém em spread_array os dados históricos de valores de spread para o par selecionado ativo-período na quantidade especificada. Deve ser notado que a ordenação dos elementos é do presente para o passado, isto é, a posição de início 0 significa a barra corrente.



Ao copiar uma quantidade desconhecida de dados, é recomendável usar um array dinâmico como array destino, porque se a quantidade solicitada de dados for menor (ou maior) que o comprimento do array destino, a função tenta realocar a memória de qual forma que os dados solicitados caibam completamente.

Se você souber a quantidade de dados que você precisa para copiar, é melhor usar um buffer alocado estaticamente, a fim de evitar a alocação de memória excessiva.

Não importa qual seja a propriedade do array destino - como series=true ou como series=false. Os dados serão copiados de tal maneira que o elemento mais antigo será localizado no início da memória física alocada para o array. Exitem 3 variantes de chamada da função.

Chamar pela posição primeira e o número de elementos requeridos

```
int  CopySpread(
   string           symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES  timeframe,        // período
   int              start_pos,        // posição de início
   int              count,            // quantidade de dados para copiar
   int              spread_array[]    // array destino para valores de spread
   );
```

Chamar pela data de início e o número de elementos requeridos

```
int  CopySpread(
   string           symbol_name,      // nome do ativo
   ENUM_TIMEFRAMES  timeframe,        // período
   datetime         start_time,       // data e hora de início
   int              count,            // quantidade de dados para copiar
   int              spread_array[]    // array destino para valores de spread
   );
```

Chamar pelas datas de início e término de um intervalo de tempo requerido

```
int  CopySpread(
```

© 2000-2024, MetaQuotes Ltd.

```
   string           symbol_name,       // nome do ativo
   ENUM_TIMEFRAMES  timeframe,         // período
   datetime         start_time,        // data e hora de início
   datetime         stop_time,         // data e hora de parada
   int              spread_array[]     // array destino para valores de spread
   );
```

Parâmetros

symbol_name

[in] Nome do ativo.

timeframe

[in] Período.

start_pos

[in] A posição de início do primeiro elemento para copiar.

count

[in] Quantidade de dados para copiar.

start_time

[in] A hora de início para o primeiro elemento para copiar.

stop_time

[in] Hora da barra, correspondente ao último elemento para copiar.

spread_array[]

[out] Array de tipo int.

Valor do Retorno

Retorna a quantidade de dados copiados ou -1 no caso de um erro.

Observação

Se o intervalo inteiro de dados solicitados não estiver disponível no servidor, a função retorna -1. Se dados fora do TERMINAL_MAXBARS (número máximo de barras no gráfico) for solicitado, a função também retornará -1.

Ao solicitar dados de um indicador, se as séries de tempo solicitadas não estiverem ainda construídas ou elas precisarem serem baixadas do servidor, a função imediatamente retornará -1, mas o processo de download/construção será iniciado.

Ao solicitar dados de um Expert Advisor ou script, o download do servidor será iniciado se o terminal não tiver estes dados localmente, ou a construção da série de tempo solicitada iniciará, se os dados puderem ser construídas a partir do histórico local mas eles não estiverem prontos ainda. A função retornará a quantidade de dados que estará pronta no momento da expiração do tempo limite, mas o download do histórico continuará, e na próximo solicitação simular, a função retornará mais dados.

Ao solicitar dados através da data de início e do número de elementos requeridos, somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de

© 2000-2024, MetaQuotes Ltd.

indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre menor ou igual ao valor especificado.

Ao solicitar dados em uma faixa especificada de datas, somente dados deste intervalo serão retornados. O intervalo é definido e contado em segundos. Isso significa que a hora de abertura de qualquer barra, para cujo valor é retornado (volume, spread, valor no buffer de indicador, preços de Abertura, Máximo, Mínimo, Fechamento ou Hora de Abertura) é sempre dentro do intervalo solicitado.

Assim, se o dia corrente for sábado, e em uma tentativa de copiar dados usando a janela de tempo semanal, você especificar *start_time=Last_Tuesday* e *stop_time=Last_Friday*, a função retornará 0, porque a hora de abertura em uma janela de tempo semanal é sempre domingo, e nenhum barra semanal cairá no intervalo especificado.

Se você precisar retornar o valor correspondente para a barra corrente incompleta, você pode usar a primeira forma de chamada, especificando *start_pos*=0 e *count*=1.

Exemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      bars=3000;
//--- buffers do indicador
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

```
                  const double &close[],
                  const long &tick_volume[],
                  const long &volume[],
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("We have received the following number of Spread values: ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Veja um exemplo de solicitação de dados históricos na seção Métodos de Vinculação de Objetos. O script disponível nesta seção mostra como obter os valores do indicador iFractals nas últimas 1000 barras e como exibir os últimos 10 fractais de alta e os últimos 10 fractais de baixa no gráfico. Uma técnica simular pode ser usada para todos os indicadores que têm dados faltantes e que são usualmente desenhados usando os seguintes estilos:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2024, MetaQuotes Ltd.

CopyTicks

A função obtém, dentro da matriz ticks_array, ticks no formato MqlTick, além disso a indexação é realizada do passado para o presente, ou seja, o tick com índice 0 é o mais antigo na matriz. Para a análise de ticks, é necessário verificar o campo *flags*, que indica o que foi alterado nesse tick.

```
int  CopyTicks(
   string          symbol_name,          // nome do símbolo
   MqlTick&        ticks_array[],        // matriz para recebimento de ticks
   uint            flags=COPY_TICKS_ALL, // sinalizador que define o tipo de ticks c
   ulong           from=0,               // data a partir da qual são solicitados os
   uint            count=0               // número de ticks que é necessário obter
   );
```

Parâmetros

symbol_name

[in] Símbolo.

ticks_array

[out] Matriz do tipo MqlTick para recebimento de ticks.

flags

[in] sinalizador que especifica o tipo de ticks solicitados. COPY_TICKS_INFO - ticks chamados pelas alterações do Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações em Last e Volume, COPY_TICKS_ALL - todos os ticks. Em qualquer tipo de solicitação, nos restantes campos da estrutura MqlTick são acrescentados os valores do tick anterior.

from

[in] Data a partir da qual são solicitados os ticks. É especificada em milissegundos desde 01.01.1970. Se o parâmetro *from*=0, são entregues as últimas *count* de ticks.

count

[in] Número de ticks solicitados. Se os parâmetros from e count não forem definidos, no matriz ticks_array[] serão registrados todos os últimos ticks disponíveis, mas não mais de 2 000.

Valor de retorno

Núerod de ticks copiados ou -1 em caso de OnCalculate() nos indicadores é chamada após a entrada de cada tick.

Nos EAs e scripts, a função CopyTicks() pode esperar até 45 segundos antes de obter o resultado: Em contraste com o indicador, cada EA e script opera em seu próprio thread e, portanto, pode esperar 45 segundos até a conclusão da sincronização. Se, durante este tempo, a quantidade necessária de ticks não forem sincronizados, CopyTicks () irá retornar ticks disponíveis, por tempo esgotado, e continuará a sincronização. A função OnTick() nos EAs não é processador de ticks, ela simplesmente notifica o EA sobre as mudanças no mercado. Essas mudanças podem ser um lote: o terminal pode fazer simultaneamente alguns ticks, mas a função OnTick () será chamada somente uma vez para notificar o EA do estado mais recente do mercado.

Velocidade de retorno de dados: o terminal armazena, para cada símbolo, os últimos 4096 ticks no cache para acesso rápido (para símbolos com livro de ofertas serão 65536 ticks), as solicitações de esses dados são feitas com maior velocidade. Ao solicitar ticks da sessão de negociação atual fora

© 2000-2024, MetaQuotes Ltd.

dos limites do cache, a CopyTicks() chama os ticks que são armazenados na memória do terminal, estas solicitações exigem mais tempo de execução. As mais lentas são as solicitações de ticks para outros dias, como, neste caso, os dados são lidos a partir do disco.

Exemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- Requesting 100 million ticks to be sure we receive the entire tick history
input int      getticks=100000000; // The number of required ticks
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // Contagem de tentativas
   bool    success=false;  // A flag de uma cópia bem sucedida de ticks
   MqlTick tick_array[];   // Tick recebendo o array
   MqlTick lasttick;       // Para receber os últimos dados do tick
   SymbolInfoTick(_Symbol,lasttick);
//--- Faça 3 tentativas de receber ticks
   while(attempts<3)
     {
      //--- Medição da hora de início antes de receber os ticks
      uint start=GetTickCount();
//--- Solicitando o histórico de ticks desde 1970.01.01 00:00.001 (parâmetro para=1 ms
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- Mostrando informações sobre o número de ticks e tempo gasto
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- Se o histórico de ticks é sincronizado, o código de erro é igual a zero
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: Ticks ainda não estão sincronizados, %d ticks recebidos p
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- Contagem de tentativas
      attempts++;
      //--- Uma pausa de um segundo para aguardar o fim da sincronização da base de da
      Sleep(1000);
     }
```

```
//--- A recepção de ticks solicitados no início do histórico de ticks falhou em três t
   if(!success)
     {
      PrintFormat("Error! Falha em receber %d ticks do %s em três tentativas",getticks
      return;
     }
   int ticks=ArraySize(tick_array);
//--- Mostrando a hora da primeira marca no array
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("Horário do último tick = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- Obtivemos o último tick no array
   datetime lastticktime=tick_array[0].time;
   PrintFormat("Horário do primeiro tick = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*100
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) failed, error %d
                  _Symbol,TimeToString(startday),TimeToString(endday),GetLastError());
      return;
     }
   ticks=MathMax(100,ticks);
//--- Mostrando os primeiros 100 ticks do último dia
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- Mostrando as primeiras 100 ofertas do último dia
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
```

© 2000-2024, MetaQuotes Ltd.

```
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) || ((tick_array[i].fl
           {
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
           }
        }
     }
  }
//+------------------------------------------------------------------+
//| Retorna a descrição de string de um tick                         |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03d ",
                            TimeToString(tick.time),tick.time_msc%1000);
//--- Checando flags
   bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
   bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
   bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
   bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
   bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
   bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- Verificar flags de negociação num primeiro tick
   if(buy_tick || sell_tick)
     {
      //--- Formando uma saída para o tick de negociação
      desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d ",tick.last,tick.v
      desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+"(Trade tick)";
     }
   else
     {
      //--- Forme uma saída diferente para um tick de informação
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
      desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
      desc=desc+"(Info tick)";
     }
//--- Retornando descrição do tick
   return desc;
  }
//+------------------------------------------------------------------+
/* Exemplo de saída
Si-12.16: recebeu 11048387 ticks em 4937 ms
```

© 2000-2024, MetaQuotes Ltd.

```
Horário do último tick = 2016.09.26 18:32:59.775
Horário do primeiro tick = 2015.06.18 09:45:01.000
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

Veja também

SymbolInfoTick, Estrutura para recebimento de preços atuais, OnTick()

© 2000-2024, MetaQuotes Ltd.

CopyTicksRange

A função recebe, na matriz ticks_array, ticks em formato MqlTick, no intervalo de datas especificado. Além disso, a indexação é realizada do passado para o presente, ou seja, o tick com índice 0 é o mais antigo na matriz. Para analisar o tick, é necessário verificar o campo *flags*, ele notifica sobre as alterações levadas a cabo.

```
int  CopyTicksRange(
   const string     symbol_name,           // nome do símbolo
   MqlTick&         ticks_array[],         // matriz para recebimento de ticks
   uint             flags=COPY_TICKS_ALL,  // sinalizador que define o tipo de ticks c
   ulong            from_msc=0,            // data a partir da qual são solicitados os
   ulong            to_msc=0               // data em que são solicitados os ticks
   );
```

Parâmetros

symbol_name

[in] Símbolo.

ticks_array

[out] Matriz estática ou dinâmica MqlTick para recebimento de ticks. Se na matriz estática não caberem todos os ticks do intervalo solicitado, serão obtidos tantos ticks quanto caberem na matriz. Além disso, a função gerará o erro ERR_HISTORY_SMALL_BUFFER (4407) .

flags

[in] sinalizador que especifica o tipo de ticks solicitados. COPY_TICKS_INFO - ticks chamados pelas alterações do Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações em Last e Volume, COPY_TICKS_ALL - todos os ticks. Em qualquer tipo de solicitação, nos restantes campos da estrutura MqlTick são acrescentados os valores do tick anterior.

from_msc

[in] Data a partir da qual são solicitados os ticks. É especificada em milissegundos desde 01.01.1970. Se o parâmetro *from_msc* não estiver especificado, são enviados os ticks a partir do início do histórico. São enviados os ticks com tempo >= from_msc.

to_msc

[in] Data na qual são solicitados os ticks. Especificado em milissegundos desde 01.01.1970. Se o parâmetro *to_msc* não estiver especificado, são enviados todos os ticks do histórico. Número de ticks solicitados. São enviados os ticks com tempo <= to_msc.

Valor de retorno

Número de ticks copiados ou -1 em caso de erro. GetLastError() pode retornar os seguintes erros:

- ERR_HISTORY_TIMEOUT - esgotado o tempo de espera de sincronização, a função enviou todo o que havia.
- ERR_HISTORY_SMALL_BUFFER - o buffer estático é muito pequeno, enviado tanto, quanto cabia na matriz.
- ERR_NOT_ENOUGH_MEMORY - não há memória suficiente para obter o histórico a partir do intervalo especificado na matriz dinâmica de ticks. Falha ao selecionar a quantidade certa de memória para uma matriz de ticks.

© 2000-2024, MetaQuotes Ltd.

Observação

A função CopyTicksRange() é usada para solicitar ticks a partir de um intervalo antigo, por exemplo, os ticks de um dia determinado no histórico. Enquanto a CopyTicks() permite definir apenas a data de início, por exemplo, obter todos os ticks desde o início do mês até o momento atual.

Veja também

SymbolInfoTick, Estrutura para recebimento de preços atuais, OnTick, CopyTicks

© 2000-2024, MetaQuotes Ltd.

iBars

Retorna o número de barras - no histórico - do símbolo e do período correspondentes.

```
int  iBars(
   const string          symbol,          // símbolo
   ENUM_TIMEFRAMES       timeframe        // período
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

Valor de retorno

Número de barras - no histórico - do símbolo e do período correspondentes, mas não mais do que o especificado nas configurações da plataforma pelo parâmetro "Número máx. de barras na janela" ("Max bars in chart")

Exemplo:

```
  Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

Veja também

Bars

© 2000-2024, MetaQuotes Ltd.

iBarShift

Busca de barra segundo o tempo. A função retorna o índice da barra na qual o tempo especificado coincide.

```
int  iBarShift(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   datetime            time,            // tempo
   bool                exact=false      // modo
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. PERIOD_CURRENT significa o período atual do gráfico atual.

time

[in] Valor do tempo para a pesquisa.

exact=false

[in] Valor retornado se a barra não for encontrada para o tempo especificado. Se *exact=false*, *iBarShift* retorna o índice da barra mais próxima cujo tempo de abertura é menor do que o especificado (*time_open<time*). Se essa barra não for encontrada (não há histórico antes do tempo especificado), a função retornará -1. Se *exact=true*, não é pesquisada a barra mais próxima e a função iBarShift retorna imediatamente -1.

Valor de retorno

Índice da barra que coincide com o tempo especificado. Se não houver barra para o tempo especificado ("buraco" no histórico), a função retorna -1 ou o índice da barra mais próxima (dependendo do parâmetro *exact*).

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- data é no domingo
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- se não houver barra para o tempo especificado, iBarShift retornará o índice da b
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- verificamos o código de erro após chamada de iBarShift()
```

© 2000-2024, MetaQuotes Ltd.

```
   int error=GetLastError();
   if(error!=0)
     {
      PrintFormat("iBarShift(): GetLastError=%d - data solicitada %s "+
                  "para %s %s no histórico disponível não encontrada",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- função iBarShift() executada com sucesso, imprimimos os resultados para exact=fa
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- precisamos encontrar o índice da barra para o tempo especificado, se ele não ex
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- função iBarShift() executada com sucesso, imprimimos os resultados para exact=tr
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Retorna o nome do dia da semana                                  |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
```

© 2000-2024, MetaQuotes Ltd.

```
   return day;
  }
//+------------------------------------------------------------------+
/* Resultado da execução
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

© 2000-2024, MetaQuotes Ltd.

iClose

Retorna o valor do preço de fechamento da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
double  iClose(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do preço de fechamento da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
}
```

Veja também

CopyClose, CopyRates

© 2000-2024, MetaQuotes Ltd.

iHigh

Retorna o valor do preço máximo da barra (especificado pelo parâmetro shift) do gráfico correspondente.

```
double  iHigh(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do preço máximo da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
}
```

Veja também

CopyHigh, CopyRates

© 2000-2024, MetaQuotes Ltd.

iHighest

Retorna o índice do maior valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente.

```
int  iHighest(
   const string        symbol,              // símbolo
   ENUM_TIMEFRAMES     timeframe,           // período
   ENUM_SERIESMODE     type,                // identificador de timeseries
   int                 count=WHOLE_ARRAY,   // número de itens
   int                 start=0              // índice
  );
```

Parâmetros

symbol

[in] Símbolo em que será realizada a pesquisa. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

type

[in] Identificador de timeseries em que será realizada a pesquisa. Pode ser qualquer um dos valores ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Número de itens de timeseries (na direção tanto da barra atual quanto do aumento do índice), entre os quais deve ser realizada a pesquisa.

start=0

[in] Índice (deslocamento em relação à barra atual) da barra inicial a partir do qual começa a pesquisa do maior valor. Valores negativos são ignorados e substituídos por um valor zero.

Valor de retorno

Índice do maior valor encontrado (deslocamento em relação à barra atual) do gráfico correspondente ou -1 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Exemplo:

```
   double val;
//--- cálculo do valor máximo do preço Close em 20 barras consecutivas
//--- a partir do índice 4 e terminando no índice 23, inclusive no gráfico atual
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("Erro ao chamar iHighest(). Código de erro=%d",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

iLow

Retorna o valor do preço mínimo da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
double  iLow(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do preço mínimo da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Veja também

CopyLow, CopyRates

© 2000-2024, MetaQuotes Ltd.

iLowest

Retorna o índice do menor valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente.

```
int  iLowest(
   const string        symbol,              // símbolo
   ENUM_TIMEFRAMES     timeframe,           // período
   ENUM_SERIESMODE     type,                // identificador de timeseries
   int                 count=WHOLE_ARRAY,   // número de itens
   int                 start=0              // índice
  );
```

Parâmetros

symbol

[in] Símbolo em que será realizada a pesquisa. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 indica o período do gráfico atual.

type

[in] Identificador de timeseries em que será realizada a pesquisa. Pode ser qualquer um dos valores ENUM_SERIESMODE.

count=WHOLE_ARRAY

[in] Número de itens de timeseries (na direção tanto da barra atual quanto do aumento do índice), entre os quais deve ser realizada a pesquisa.

start=0

[in] Índice (deslocamento em relação à barra atual) da barra inicial a partir do qual começa a busca do menor valor. Valores negativos são ignorados e substituídos por um valor zero.

Valor de retorno

Índice do menor valor encontrado (deslocamento em relação à barra atual) do gráfico correspondente ou -1 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Exemplo:

```
   double val;
//--- pesquisa da barra com um valor mínimo do volume real em 15 barras consecutivas
//--- a partir do índice 10 e terminando no índice 24, inclusive no gráfico atual
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("Erro ao chamar iLowest(). Código de erro=%d",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

iOpen

Retorna o valor do preço de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
double  iOpen(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do preço de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
}
```

Veja também

CopyOpen, CopyRates

© 2000-2024, MetaQuotes Ltd.

iTime

Retorna o valor do tempo de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
datetime  iTime(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do tempo de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- data é no domingo
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- se não houver barra para o tempo especificado, iBarShift retornará o índice da b
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
```

© 2000-2024, MetaQuotes Ltd.

```
                symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
                bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- precisamos encontrar o índice da barra para o tempo especificado, se ele não ex
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
                symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
  }
//+------------------------------------------------------------------+
//| Retorna o nome do dia da semana                                  |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
/* Resultado:
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

Veja também

CopyTime, CopyRates

© 2000-2024, MetaQuotes Ltd.

iTickVolume

Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente.

```
long  iTickVolume(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do volume de tick da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Veja também

CopyTickVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iRealVolume

Retorna o valor do volume real da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
long  iRealVolume(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

O valor do volume real da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Veja também

CopyRealVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iVolume

Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente.

```
long  iVolume(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do volume de tick da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Veja também

CopyTickVolume, CopyRates

© 2000-2024, MetaQuotes Ltd.

iSpread

Retorna o valor do spread da barra (indicado pelo parâmetro shift) do gráfico correspondente.

```
long  iSpread(
   const string        symbol,          // símbolo
   ENUM_TIMEFRAMES     timeframe,       // período
   int                 shift            // deslocamento
   );
```

Parâmetros

symbol

[in] Nome simbólico do instrumento. NULL significa o símbolo atual.

timeframe

[in] Período. Pode ser um dos valores da enumeração ENUM_TIMEFRAMES. 0 significa o período do gráfico atual.

shift

[in] Índice do valor recebido do timeseries (deslocamento - número especificado de barras atrás - em relação à barra atual).

Valor de retorno

Valor do spread da barra (indicado pelo parâmetro shift) do gráfico correspondente ou 0 no caso de erro. Para mais informações sobre o erro, deve-se chamar a função GetLastError().

Observação

A função sempre retorna dados relevantes, para isso, a cada chamada, faz uma solicitação aos timeseries para o símbolo/período especificado. Isso significa que, se não houver dados prontos na primeira chamada de função, preparar o resultado poderá levar algum tempo.

A função não armazena os resultados de chamadas anteriores, pois não há cache local para retorno rápido do valor.

Exemplo:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Function-event handler "tick"                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
           "Open: "  ,DoubleToString(open,Digits()),"\n",
           "High: "  ,DoubleToString(high,Digits()),"\n",
           "Low: "   ,DoubleToString(low,Digits()),"\n",
           "Close: " ,DoubleToString(close,Digits()),"\n",
           "Volume: ",IntegerToString(volume),"\n",
           "Bars: "  ,IntegerToString(bars),"\n"
           );
  }
```

Veja também

CopySpread, CopyRates

© 2000-2024, MetaQuotes Ltd.

Símbolos personalizados

Funções para criar e editar as propriedades dos símbolos personalizados.

Ao conectar o terminal a um determinado servidor de negociação, o usuário pode trabalhar com os TimeSeries dos instrumentos financeiros que fornece a corretora. A lista de símbolos - mostrada na janela da Observação do mercado - exibe os instrumentos financeiros disponíveis, e um grupo separado de funções permite obter informações sobre as propriedades do símbolo, sessões de negociação e atualizações do livro de ofertas.

O grupo de funções nesta seção permite criar seus próprios símbolos personalizados. Para fazer isso, é possível criar os símbolos existentes do servidor de negociação, arquivos de texto ou históricos de dados externos.

Função	Ação
CustomSymbolCreate	Cria um símbolo personalizado com o nome especificado no grupo definido
CustomSymbolDelete	Remove o símbolo personalizado com o nome especificado
CustomSymbolSetInteger	Define o valor do tipo inteiro para o símbolo personalizado
CustomSymbolSetDouble	Define o valor do tipo real para o símbolo personalizado
CustomSymbolSetString	Define o valor do tipo cadeia de caracteres para o símbolo personalizado
CustomSymbolSetMarginRate	Define o coeficiente de cobrança da margem - dependendo do tipo e direção da ordem - para o símbolo personalizado
CustomSymbolSetSessionQuote	Define a hora de início e fim da sessão de cotação especificada para o símbolo especificado e dia da semana
CustomSymbolSetSessionTrade	Define a hora de início e fim da sessão de negociação especificada para o símbolo especificado e dia da semana
CustomRatesDelete	Exclui todas as barras no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado
CustomRatesReplace	Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlRates, no intervalo de tempo definido
CustomRatesUpdate	Adiciona ao histórico do instrumento personalizado as barras que faltam e substitui os dados existentes na matriz do tipo MqlRates
CustomTicksAdd	Adds data from an array of the MqlTick type to the price history of a custom symbol. The custom symbol must be selected in the Market Watch window.
CustomTicksDelete	Exclui todos os ticks no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado

© 2000-2024, MetaQuotes Ltd.

Função	Ação
CustomTicksReplace	Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlTick, no intervalo de tempo definido
CustomBookAdd	Envia o estado do livro de ofertas do instrumento personalizado

© 2000-2024, MetaQuotes Ltd.

CustomSymbolCreate

Cria um símbolo personalizado com o nome especificado no grupo definido.

```
bool  CustomSymbolCreate(
   const string     symbol_name,          // nome do símbolo personalizado
   const string     symbol_path="",       // nome do grupo em que será criado o símbolo
   const string     symbol_origin=NULL    // nome do símbolo com base no qual será cria
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado. Ele não deve conter grupos ou sub-grupos em que se encontre o símbolo.

symbol_path=""

[in] Nome do grupo em que é criado o símbolo.

symbol_origin=NULL

[in] Nome do símbolo a partir do qual serão copiadas as propriedades do símbolo personalizado gerado. Depois de criar um símbolo personalizado, você pode alterar qualquer propriedade para o valor desejado com as funções correspondentes.

Valor retornado

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Todos os símbolos personalizados são gerados na seção especial Custom. Se o nome de grupo não for especificado (o parâmetro *symbol_path* na função CustomSymbolCreate contém uma string vazia ou NULL), o símbolo personalizado será criado na raiz da seção Custom. Aqui você pode fazer uma analogia com o sistema de arquivos, onde grupos e subgrupos podem ser vistos como pastas e subpastas

O nome do símbolo e descrição do grupo são definidos apenas por caracteres latinos sem pontuação, espaços e caracteres especiais (são permitidos ".", "_", "&" e "#"). Não é recomendado usar - nos nomes dos símbolos - os caracteres <, >, :, ", /, |, ?, *.

O nome do símbolo personalizado deve ser único, independentemente do nome do grupo em que ele é criado. Se já existir um símbolo com o mesmo nome, a função CustomSymbolCreate() retornará false, enquanto a chamada subsequente de GetLastError() gerará o erro 5300 (ERR_NOT_CUSTOM_SYMBOL) ou 5304 (ERR_CUSTOM_SYMBOL_EXIST).

O nome do símbolo não deve exceder 31 caracteres, caso contrário, CustomSymbolCreate() retornará false e será gerado o ero 5302 – ERR_CUSTOM_SYMBOL_NAME_LONG.

Parâmetro *symbol_path* pode ser definido de duas maneiras:

- apenas o nome do grupo sem o nome do símbolo personalizado, por exemplo – "CFD\\Metals". É melhor usar essa opção para evitar erros.
- ou nome de <grupo> + separador de grupo "\\"+<nome de símbolo personalizado>, por exemplo – "CFD\\Metals\\Platinum". Nesse caso, o nome do grupo deve terminar com o nome exato do

© 2000-2024, MetaQuotes Ltd.

símbolo personalizado. Mesmo em caso de incompatibilidade, o símbolo personalizado será criado, mas não no grupo pretendido. Por exemplo, se *symbol_path*="CFD\\Metals**P**latinum" e *symbol_name*="**p**latinum" (erro no registro), será criado o símbolo personalizado com o nome "platinum" no grupo "Custom\CFD\Metals\Platinum". Nesse caso, a função SymbolInfoGetString("platinum",SYMBOL_PATH) retornará o valor "Custom\CFD\Metals\Platinum\platinum".

Deve-se ter em mente que a propriedade SYMBOL_PATH retorna o caminho com o nome do símbolo no final. Portanto, você não pode simplesmente copiá-lo sem alterações, se quiser criar um símbolo personalizado exatamente no mesmo grupo. Nesse caso, é necessário cortar o nome do símbolo para não obter o resultado descrito acima.

Se como parâmetro *symbol_origin* for especificado um símbolo inexistente, o símbolo personalizado será criado vazio, como se o parâmetro *symbol_origin* não tivesse sido especificado. Nesse caso, será gerado o erro 4301 – ERR_MARKET_UNKNOWN_SYMBOL.

O comprimento do parâmetro *symbol_path* não deve exceder 127 caracteres considerando "Custom\\", separador de grupo "\\" e nome do símbolo se for indicado no final.

Veja também

SymbolName, SymbolSelect, CustomSymbolDelete

© 2000-2024, MetaQuotes Ltd.

CustomSymbolDelete

Remove o símbolo personalizado com o nome especificado.

```
bool  CustomSymbolDelete(
   const string     symbol_name          // nome do símbolo personalizado
   );
```

Parâmetros

symbol

[in] Nome do símbolo personalizado. Não deve corresponder ao nome do símbolo existente.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

O símbolo personalizado que aparece na observação do mercado (Market Watch) ou que abre o gráfico não pode ser excluído.

Veja também

SymbolName, SymbolSelect, CustomSymbolCreate

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetInteger

Define o valor do tipo inteiro para o símbolo personalizado.

```
bool  CustomSymbolSetInteger(
   const string              symbol_name,       // nome do símbolo
   ENUM_SYMBOL_INFO_INTEGER  property_id,       // identificador de propriedade
   long                      property_value     // valor de propriedade
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

property_id

[in] Identificador da propriedade do símbolo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_INTEGER.

property_value

[in] Variável de tipo long contendo o valor da propriedade.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Histórico de minuto e de ticks do símbolo personalizado é completamente removido, se, na especificação do símbolo, for alterada qualquer uma destas propriedades:

- SYMBOL_CHART_MODE – tipo de preço usado para plotagem de barras (Bid ou Last)
- SYMBOL_DIGITS – número de casas decimais para exibição do preço

Após a remoção do histórico personalizado, o terminal tentará criar um novo histórico usando as propriedades atualizadas. A mesma coisa acontece quando você altera manualmente as propriedades do símbolo personalizado.

Veja também

SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetDouble

Define o valor do tipo real para o símbolo personalizado.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,       // nome do símbolo
   ENUM_SYMBOL_INFO_DOUBLE   property_id,       // identificador de propriedade
   double                    property_value     // valor da propriedade
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

property_id

[in] Identificador da propriedade do símbolo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_DOUBLE.

property_value

[in] Variável de tipo double contendo o valor da propriedade.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Histórico de minuto e de ticks do símbolo personalizado é completamente removido, se, na especificação do símbolo, for alterada qualquer uma destas propriedades:

- SYMBOL_POINT - valor de um ponto
- SYMBOL_TRADE_TICK_SIZE - valor do tick que define o variação mínima permitida no preço
- SYMBOL_TRADE_TICK_VALUE - custo quado o preço varia um tick para posição lucrativa

Após a remoção do histórico personalizado, o terminal tentará criar um novo histórico usando as propriedades atualizadas. A mesma coisa acontece quando você altera manualmente as propriedades do símbolo personalizado.

Veja também

SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetString

Define o valor do tipo cadeia de caracteres para o símbolo personalizado.

```
bool  CustomSymbolSetString(
   const string              symbol_name,       // nome do símbolo
   ENUM_SYMBOL_INFO_STRING   property_id,       // identificador de propriedade
   string                    property_value     // valor de propriedade
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

property_id

[in] Identificador da propriedade do símbolo. O valor pode ser um dos valores da enumeração ENUM_SYMBOL_INFO_STRING.

property_value

[in] Variável de tipo string contendo o valor da propriedade.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Histórico de minuto e de ticks do símbolo personalizado é completamente removido, se, na especificação do símbolo, for alterada a propriedade SYMBOL_FORMULA, ela define a fórmula para plotagem do preço do símbolo personalizado. Após a remoção do histórico personalizado, o terminal tentará criar um novo histórico com base na nova fórmula. A mesma coisa acontece quando você altera manualmente a fórmula do símbolo personalizado.

Veja também

SymbolInfoString

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetMarginRate

Define o coeficiente de cobrança da margem - dependendo do tipo e direção da ordem - para o símbolo personalizado.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // nome do símbolo
   ENUM_ORDER_TYPE    order_type,               // tipo de ordem
   double             initial_margin_rate,      // coeficiente de cobrança da margem i
   double             maintenance_margin_rate   // coeficiente de cobrança da margem d
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

order_type

[in] Tipo de ordem.

initial_margin_rate

[in] Variável de tipo double com o valor de coeficiente de cobrança da margem inicial. A margem inicial é o tamanho da soma de garantia (margem) das transações concluídas com volume de um lote para a direção correspondente. Ao multiplicar o coeficiente pela margem inicial, nós podemos obter o montante que será reservado na conta ao colocar a ordem do tipo especificado.

maintenance_margin_rate

[in] Variável do tipo double com o valor de coeficiente de cobrança da margem de manutenção. A margem de manutenção é a quantidade mínima para manter uma posição aberta com volume de um lote para a direção correspondente. Ao multiplicar o coeficiente pela margem de manutenção, nós podemos obter o montante que será reservado na conta após a ativação da ordem do tipo especificado.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Veja também

SymbolInfoMarginRate

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

Define a hora de início e fim da sessão de cotação especificada para o símbolo especificado e dia da semana.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // nome do símbolo
   ENUM_DAY_OF_WEEK  day_of_week,          // dia da semana
   uint              session_index,        // número de sessão
   datetime          from,                 // hora de início da sessão
   datetime          to                    // hora de fim da sessão
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

ENUM_DAY_OF_WEEK

[in] Dia da semana toma um valor da enumeração ENUM_DAY_OF_WEEK.

uint

[in] Número de ordem da sessão para a qual é necessário definir a hora de início e fim. Indexação de sessões começa por 0.

from

[in] Hora de início de sessão em segundos - a partir de 00:00 -, o valor da data será ignorado na variável.

to

[in] Hora de fim de sessão em segundos - a partir de 00:00 -, o valor da data será ignorado na variável.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Se a sessão com o *session_index* definido já existir, a função simplesmente editará o início e fim da sessão.

Se transferidos parâmetros nulos de início e fim para a sessão, isto é, especificados *from=0* e *to=0*, a sessão correspondente com índice *session_index* será removida, enquanto a numeração das sessões será deslocada para baixo.

É possível adicionar sessões só sequencialmente, isto é, a sessão com índice *session_index=1* pode ser adicionada apenas se já existir uma sessão com um índice igual a 0. Ao violar esta regra, não é criada uma nova sessão, e a função retornará false.

Veja também

SymbolInfoSessionQuote, Informações sobre o instrumento , TimeToStruct, Estrutura de data

© 2000-2024, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

Define a hora de início e fim da sessão de negociação especificada para o símbolo especificado e dia da semana.

```
bool  CustomSymbolSetSessionTrade(
   constg            symbol_name,         // nome do símbolo
   ENUM_DAY_OF_WEEK  day_of_week,         // dia da semana
   uint              session_index,       // número de sessão
   datetime          from,                // hora de início da sessão
   datetime          to                   // hora de fim da sessão
   );
```

Parâmetros

symbol_name

[in] Nome do símbolo personalizado.

ENUM_DAY_OF_WEEK

[in] Dia da semana toma um valor da enumeração ENUM_DAY_OF_WEEK.

uint

[in] Número de ordem da sessão para a qual é necessário definir a hora de início e fim. Indexação de sessões começa por 0.

from

[in] Hora de início de sessão em segundos - a partir de 00:00 -, o valor da data será ignorado na variável.

to

[in] Hora de fim de sessão em segundos - a partir de 00:00 -, o valor da data será ignorado na variável.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Observação

Se a sessão com o *session_index* definido já existir, a função simplesmente editará o início e fim da sessão.

Se transferidos parâmetros nulos de início e fim para a sessão, isto é, especificados *from=0* e *to=0*, a sessão correspondente com índice *session_index* será removida, enquanto a numeração das sessões será deslocada para baixo.

É possível adicionar sessões só sequencialmente, isto é, a sessão com índice *session_index=1* pode ser adicionada apenas se já existir uma sessão com um índice igual a 0. Ao violar esta regra, não é criada uma nova sessão, e a função retornará false.

Veja também

SymbolInfoSessionTrade, Informações sobre o instrumento, TimeToStruct, Estrutura de data

© 2000-2024, MetaQuotes Ltd.

CustomRatesDelete

Exclui todas as barras no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado.

```
int  CustomRatesDelete(
   const string     symbol,        // nome do símbolo
   datetime         from,          // a partir de qual data
   datetime         to             // até qual data
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

from

[in] Tempo da primeira barra no histórico de preço, na faixa definida a ser removida.

to

[in] Tempo da última barra no histórico de preço, na faixa definida a ser removida.

Valor de retorno

Número de barras removidas ou -1 em caso de erro.

Veja também

CustomRatesReplace, CustomRatesUpdate, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomRatesReplace

Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlRates, no intervalo de tempo definido.

```
int  CustomRatesReplace(
   const string     symbol,             // nome do símbolo
   datetime         from,               // a partir de qual data
   datetime         to,                 // até qual data
   const MqlRates&  rates[],            // matriz com dados que é necessário aplicar a
   uint             count=WHOLE_ARRAY   // número de elementos a serem usados na matri
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

from

[in] Tempo da primeira barra no histórico de preço, na faixa definida a ser atualizada.

to

[in] Tempo da última barra no histórico de preço, na faixa definida a ser atualizada.

rates[]

[in] Matriz de dados históricos do tipo MqlRates para o período M1.

count=WHOLE_ARRAY

[in] Número de elementos da matriz *rates[]* a serem usados para substituição. O valor WHOLE_ARRAY indica que para substituição é necessário usar todos os elementos da matriz *rates[]*.

Valor de retorno

Número de barras atualizadas ou -1 em caso de erro.

Geral

Se a barra na matriz *rates[]* ficar fora do intervalo definido, ela será ignorada. Se essa barra existir no histórico de preço e ficar dentro do intervalo definido, ela será substituída. Todas as outras barras - no histórico de preço atual - fora do intervalo definido permanecem inalteradas. Os dados da matriz *rates[]* devem estar corretas em relação aos preços OHLC, enquanto o tempo de abertura de barras deve corresponder ao período M1.

Veja também

CustomRatesDelete, CustomRatesUpdate, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomRatesUpdate

Adiciona ao histórico do instrumento personalizado as barras que faltam e substitui os dados existentes na matriz do tipo MqlRates.

```
int  CustomRatesUpdate(
   const string     symbol,             // nome do símbolo personalizado
   const MqlRates&  rates[],            // matriz com dados que devem ser aplicadas ao
   uint             count=WHOLE_ARRAY   // número de elementos a serem usados na matri
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

rates[]

[in] Matriz de dados históricos do tipo MqlRates para o período M1.

count=WHOLE_ARRAY

[in] Número de elementos da matriz *rates[]* a serem usados para atualização. O valor WHOLE_ARRAY indica que é necessário usar todos os elementos da matriz *rates[]*.

Valor de retorno

Número de barras atualizadas ou -1 em caso de erro.

Geral

Se a barra na matriz *rates[]* não existir no histórico atual do instrumento personalizado, ela será adicionada. Se essa barra já existir, ela será é substituída. Todas as outras barras no histórico de preço atual permanecem inalteradas. Os dados da matriz *rates[]* devem estar corretas em relação aos preços OHLC, enquanto o tempo de abertura de barras deve corresponder ao período M1.

Veja também

CustomRatesReplace, CustomRatesDelete, CopyRates

© 2000-2024, MetaQuotes Ltd.

CustomTicksAdd

Adds data from an array of the MqlTick type to the price history of a custom symbol. The custom symbol must be selected in the Market Watch window.

```
bool  CustomTicksAdd(
   const string     symbol,             // Symbol name
   const MqlTick&   ticks[],            // The array with tick data that should be app
   uint             count=WHOLE_ARRAY   // número de elementos a serem usados a parti
   );
```

Parameters

symbol

[in] The name of the custom symbol.

ticks[]

[in] An array of tick data of the MqlTick type arranged in order of time from earlier data to more recent ones, i.e. ticks[k].time_msc <= ticks[n].time_msc, if k<n.

count=WHOLE_ARRAY

[in] Número de elementos da matriz *ticks[]* a serem utilizados para adição. O valor WHOLE_ARRAY indica que é necessário usar todos os elementos da matriz *ticks[]*.

Return Value

Número de ticks adicionados ou -1 em caso de erro.

Further Note

The CustomTicksAdd function only works for custom symbols opened in the Market Watch window. If the symbol is not selected in Market Watch, then you should add ticks using CustomTicksReplace.

A função CustomTicksAdd permite transmitir ticks como se chegassem desde o servidor da corretora. Os dados não são salvos diretamente no banco de dados de ticks, em vez disso, são enviados para a janela "Observação do mercado". É a partir dela que o terminal armazena os ticks em sua base. Quando a quantidade de dados transferidos por chamada é grande, a função altera seu comportamento para poupar recursos. Se forem transferidos mais de 256 ticks, os dados serão divididos em duas partes. A primeira parte (grande) é imediatamente carregada diretamente na base de ticks (como faz a CustomTicksReplace). A segunda parte, que consiste nos últimos 128 ticks, é transferida para a janela "Observação do mercado" e, em seguida, é salva pelo terminal no banco de dados de ticks.

A estrutura MqlTick possui dois valores de campo com valores temporais, isto é, time (hora do tick em segundos) e time_msc (hora de tick em milissegundos). Eles realizam o cálculo desde 01 de janeiro de 1970. Nos ticks adicionados, o processamento destes campos é realizado acordo com as seguintes regras na ordem estabelecida:

1. se o valor for ticks[k].time_msc!=0, usamo-lo para preenchimento do campo ticks[k].time, quer dizer, para o tick é definido o tempo ticks[k].time=ticks[k].time_msc/1000 (divisão inteira)
2. se ticks[k].time_msc==0 e ticks[k].time!=0, a hora em milissegundos é obtida pela multiplicação por 1000, isto é, ticks[k].time_msc=ticks[k].time*1000
3. se ticks[k].time_msc==0 e ticks[k].time==0, nestes campos é registrado o tempo atual do servidor de negociação em milissegundos quando chamada a funçãoCustomTicksAdd.

© 2000-2024, MetaQuotes Ltd.

Se o valor dos campos ticks[k].bid, ticks[k].ask, ticks[k].last ou ticks[k].volume for maior do que zero, no campo ticks[k].flags, é escrita a combinação dos sinalizadores correspondentes:

- TICK_FLAG_BID – tick alterou o preço Bid
- TICK_FLAG_ASK – tick alterou o preço Ask
- TICK_FLAG_LAST – tick alterou o preço da última transação
- TICK_FLAG_VOLUME – tick mudou o volume

Se o valor de campo é menor ou igual a zero, o sinalizador correspondente não é gravado nos campo ticks[k].flags.

Sinalizadores TICK_FLAG_BUY e TICK_FLAG_SELL não são adicionados no histórico do instrumento personalizado.

See also

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomTicksDelete

Exclui todos os ticks no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado.

```
int  CustomTicksDelete(
   const string     symbol,            // nome do símbolo
   long             from_msc,          // a partir de qual data
   long             to_msc             // até qual data
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

from_msc

[in] Tempo do primeiro tick no histórico de preço, na faixa definida a ser removida. Tempo em milissegundos desde 01.01.1970.

to_msc

[in] Tempo do último tick no histórico de preço, na faixa definida a ser removida. Tempo em milissegundos desde 01.01.1970.

Valor de retorno

Número de ticks removidos ou -1 em caso de erro.

Veja também

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomTicksReplace

Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlTick, no intervalo de tempo definido.

```
int  CustomTicksReplace(
   const string     symbol,            // nome do símbolo
   datetime         from_msc,          // a partir de qual data
   datetime         to_msc,            // até qual data
   const MqlTick&   ticks[],           // matriz com dados de tick que se devem aplica
   uint             count=WHOLE_ARRAY  // número de elementos a serem usados a partir
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

from_msc

[in] Tempo do primeiro tick no histórico de preço, na faixa definida a ser removida. Tempo em milissegundos desde 01.01.1970.

to_msc

[in] Tempo do último tick no histórico de preço, na faixa definida a ser removida. Tempo em milissegundos desde 01.01.1970.

ticks[]

[in] Matriz dos dados de tick - de tipo MqlTick - classificados segundo o tempo em ordem crescente.

count=WHOLE_ARRAY

[in] Número de elementos da matriz *ticks[]* que será usado para substituir no intervalo de tempo especificado. O valor WHOLE_ARRAY indica que é necessário usar todos os elementos da matriz *ticks[]*.

Valor de retorno

true, em caso de sucesso, caso contrário, false. Para obter informações sobre o erro, você deve chamar a função GetLastError().

Geral

Como, no canal de cotações, muitas vezes alguns ticks podem ter o mesmo tempo com uma precisão de milissegundos (a hora do tick é armazenado no campo *time_msc* da estrutura MqlTick), a função CustomTicksReplace não classifica automaticamente os elementos da matriz *ticks[]* de acordo com o tempo. Portanto, a matriz de ticks deve ser pré-ordenada por tempo ascendente.

Os ticks são substituídos sequencialmente, dia após dia, até a hora especificada em *to_msc* ou até que ocorra um erro. Primeiro, é processado o primeiro dia da faixa especificada, em seguida, o seguinte, e assim por diante. Assim que encontrada uma discrepância no tempo do tick quanto à ordem crescente (não decrescente), é interrompido o processo de substituição de ticks para o dia atual. Ao acontecer isto, os ticks do dia anterior são substituídos com sucesso, enquanto o dia atual

© 2000-2024, MetaQuotes Ltd.

(no momento do tick errado) e todos os dias restantes no intervalo definido permanecerão inalterados.

Se, na matriz *ticks[]*, não existirem dados de tick um certo dia (de um modo geral, um intervalo de qualquer duração), após serem implementados os ticks da `ticks[]`, no histórico do instrumento personalizado é formado um "buraco", que corresponde aos dados em falta. Em outras palavras, a chamada de CustomTicksReplace com os ticks em falta para uma determinada faixa será equivalente à remoção da parte do histórico de negociação, como se fosse chamado CustomTicksDelete com um intervalo "buracado".

Se não houver dados no banco de dados de ticks no intervalo de tempo especificado, a CustomTicksReplace simplesmente adicionará os ticks da matriz ticks[] a ele.

A função CustomTicksReplace funciona diretamente com o banco de dados de ticks.

Veja também

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

© 2000-2024, MetaQuotes Ltd.

CustomBookAdd

Envia o estado do livro de ofertas do instrumento personalizado. A função permite transmitir o livro de ofertas como se ele viesse do servidor da corretora.

```
int  CustomBookAdd(
   const string        symbol,            // nome do símbolo
   const MqlBookInfo&  books[]            // array com as descrições dos elementos do
   uint                count=WHOLE_ARRAY  // número de elementos a serem usados
   );
```

Parâmetros

symbol

[in] Nome do instrumento personalizado.

books[]

[in] Array de dados do tipo MqlBookInfo descrevendo completamente o estado do livro de ofertas — todos os pedidos de compra e de venda. O estado de livro de ofertas enviado substitui completamente o anterior.

count=WHOLE_ARRAY

[in] Número do elementos do array *books* que deve ser enviado para a função. Por padrão, é usado o array inteiro.

Valor retornado

Número de ticks adicionados ou -1 no caso de erro.

Observação

Função CustomBookAdd funciona apenas para símbolos personalizados em que o livro de ofertas está aberto — por meio da interface da plataforma ou função MarketBookAdd.

Ao adicionar ao livro de ofertas, os preços Bid e Ask do instrumento não são atualizados. Você deve controlar de forma independente a mudança dos melhores preços e adicionar ticks com a ajuda de CustomTicksAdd.

É verificada a exatidão dos dados transmitidos: para cada elemento devem ser indicados tipo, preço e volume. Além disso, MqlBookInfo.volume e MqlBookInfo.volume_real não devem ser zero ou negativos, se ambos os volumes forem negativos, isso será considerado um erro. Você pode especificar qualquer um dos volumes ou ambos, sendo que será tomado aquele que é indicado ou é positivo:

```
volume=-1 && volume_real=2 — será usado volume_real=2,
a volume=3 && volume_real=0 — será usado volume=3.
```

O volume com maior precisão MqlBookInfo.volume_real tem uma prioridade mais alta que o habitual MqlBookInfo.volume. Se, para o elemento do livro de ofertas, ambos os valores forem especificados, será usado volume_real.

A sequência de elementos MqlBookInfo no array *books* não importa. Ao salvar dados, o terminal os classifica por preço independentemente.

© 2000-2024, MetaQuotes Ltd.

Ao salvar dados, é verificado o parâmetro "Profundidade do livro de ofertas" (SYMBOL_TICKS_BOOKDEPTH) recebendo o instrumentos personalizado. Se, no livro de ofertas transmitido, o número de pedidos de venda exceder esse valor, os níveis excedentes são descartados. Da mesma forma, para pedidos de compra.

Veja como é preenchido o array *books*:

Estado do livro de ofertas	Preenchimento de books[]

Volume / Price
100.00 / 1.14337
50.00 / 1.14336
40.00 / 1.14335
10.00 / 1.14333
10.00 / 1.14322
90.00 / 1.14320
100.00 / 1.14319
10.00 / 1.14318
	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

Exemplo:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ativamos o livro de ofertas para o instrumento a partir do qual vamos tomas dado
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Tick function                                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- copiamos os preços atuais a partir de um instrumento habitual para um instrument
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Book function                                                    |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- copiamos o estado atual do livro de ofertas a partir de um instrumento habitual
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

Veja também

MarketBookAdd, CustomTicksAdd, OnBookEvent

© 2000-2024, MetaQuotes Ltd.

Operações de Gráficos

As funções para definir as propriedades do gráfico (ChartSetInteger, ChartSetDouble e ChartSetString) são assíncronas e servem para o envio de alterações para o gráfico. Quando essas funções são executadas com sucesso, o comando entra na fila geral de eventos do gráfico. A alteração das propriedades do gráfico é feita durante o processamento da fila de eventos de dado gráfico.

Por esse motivo, você não deve esperar uma atualização imediata do gráfico, após chamar funções assíncronas. Para forçar a atualização de aparência e propriedades do gráfico, use a função ChartRedraw().

Função	Ação
ChartApplyTemplate	Aplica o modelo específico de um arquivo especificado para o gráfico
ChartSaveTemplate	Salva as configurações atuais do gráfico em um modelo com um nome especificado
ChartWindowFind	Retorna o número de uma sub-janela onde um indicador é plotado
ChartTimePriceToXY	Converte as coordenadas do gráfico a partir da representação tempo/preço para as coordenadas X e Y
ChartXYToTimePrice	Converte as coordenadas X e Y no gráfico para os valores de tempo e preço
ChartOpen	Abre um novo gráfico com o símbolo e período especificados
ChartClose	Fecha um gráfico especificado
ChartFirst	Retorna o ID do primeiro gráfico do terminal do cliente
ChartNext	Retorna o ID do gráfico do próximo gráfico para um especificado
ChartSymbol	Retorna o nome do símbolo do gráfico especificado
ChartPeriod	Retorna o valor do período do gráfico especificado
ChartRedraw	Chama um redesenho forçado de um gráfico especificado
ChartSetDouble	Define o valor duplo para uma propriedade correspondente do gráfico especificado
ChartSetInteger	Define um valor inteiro (datetime, int, color, bool ou char) para uma propriedade correspondente do gráfico especificado
ChartSetString	Define o valor da sequência para uma propriedade do gráfico especificado
ChartGetDouble	Retorna a propriedade de valor duplo do gráfico especificado
ChartGetInteger	Retorna a propriedade de valor inteiro do gráfico especificado
ChartGetString	Retorna a propriedade de valor da sequência do gráfico especificado
ChartNavigate	Executa deslocamento do gráfico especificado no número especificado de barras em relação à posição definida no gráfico

© 2000-2024, MetaQuotes Ltd.

Função	Ação
ChartID	Retorna o ID do gráfico atual
ChartIndicatorAdd	Adiciona um indicador com o identificador definido a uma janela do gráfico especificado
ChartIndicatorDelete	Remove um indicador com um nome definido a partir de janela do gráfico especificado
ChartIndicatorGet	Retorna o manuseio do indicador com nome abreviado definido na janela do gráfico especificado
ChartIndicatorName	Retorna o nome abreviado do indicador pelo número na lista de indicadores sobre a janela do gráfico especificado
ChartIndicatorsTotal	Retorna o número total de indicadores utilizados para a janela do gráfico especificado
ChartWindowOnDropped	Retorna o número (índice) da sub-janela do gráfico, o Expert Advisor ou o script foi solto pelo mouse
ChartPriceOnDropped	Retorna a coordenada de preço do ponto do gráfico em que o Expert Advisor ou o script foi solto pelo mouse
ChartTimeOnDropped	Retorna a coordenada de tempo do ponto do gráfico em que o Expert Advisor ou Script foi solto pelo mouse
ChartXOnDropped	Retorna a coordenada X do ponto do gráfico em que o Expert Advisor ou script foi solto
ChartYOnDropped	Retorna a coordenada Y do ponto do gráfico em que o Expert Advisor ou script foi solto
ChartSetSymbolPeriod	Modifica o valor do símbolo e um período do gráfico especificado
ChartScreenShot	Fornece uma captura de tela do gráfico de seu estado atual em formato GIF, PNG ou BMP, dependendo da extensão especificada

© 2000-2024, MetaQuotes Ltd.

ChartApplyTemplate

Aplica-se um modelo específico de um arquivo especificado para o gráfico. O comando é adicionado para plotar uma fila de mensagens de gráfico e será executado após o processamento de todos os comandos anteriores.

```
bool  ChartApplyTemplate(
   long          chart_id,      // ID Gráfico
   const string  filename       // Nome Arquivo Modelo (Template)
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

filename

[in] O nome do arquivo contendo o modelo.

Valor do Retorno

Retorna verdadeiro se o comando foi adicionado à fila do gráfico, caso contrário será falso. Para obter uma informação sobre o erro, chamar a função GetLastError().

Observação

O Assessor Especialista (Expert Advisor) será descarregado e não poderá ser capaz de continuar operando em caso de carregamento bem sucedido de um novo modelo de gráfico que estiver anexado.

Ao aplicar o modelo ao gráfico, permissões de negociação podem ser limitadas devido a razões de segurança:

A permissão de negociação ao vivo não pode ser estendida para os Expert Advisors lançados aplicando o modelo usando a função ChartApplyTemplate().

Se o programa mql5 chamar a função ChartApplyTemplate() não possuir permissão para negociar, o Expert Advisor lançado através do modelo também não será capaz de negociar, independentemente das configurações do modelo.

Se o programa mql5 chamar a função ChartApplyTemplate() possuir permissão para negociar, enquanto não há tal permissão nas configurações do modelo, o Expert Advisor lançado através do modelo não será capaz de negociar.

Usando Modelos (Templates)

Os recursos da linguagem MQL5 permitem a configuração de propriedades de múltiplos gráficos, incluindo cores usando a função ChartSetInteger():

- Cor do fundo do gráfico;
- Cor dos eixos, escala e da linha OHLC;
- Cor da grade;
- Cor de volumes e níveis de posições abertas;

© 2000-2024, MetaQuotes Ltd.

- Cor da barra de alta, sombra e borda da barra de alta (também denominado como castiçal ou vela de alta);
- Cor da barra de baixa, sombra e borda da barra de baixa (também denominado como castiçal ou vela de baixa);
- Cor da linha do gráfico e barra Doji (também denominado como castiçal ou vela Doji);
- Cor do corpo da barra de alta (também denominado como castiçal ou vela de alta);
- Cor do corpo da barra de baixa (também denominado como castiçal ou vela de baixa);
- Cor da linha do preço de compra;
- Cor da linha do preço de venda;
- Cor da linha do último acordo de preço (Last);
- Cor dos níveis das ordens de paradas, Parar Perda e Tomar Lucro (Stop Loss e Take Profit).

Além disso, pode haver múltiplos objetos gráficos e indicadores sobre um gráfico. Você pode configurar um gráfico com todos os indicadores necessários uma única vez e salvar como um modelo (template). Este modelo salvo poderá ser aplicado em qualquer gráfico.

A função ChartApplyTemplate() é destinada ao uso de um modelo (template) salvo previamente, e pode ser usada em qualquer programa MQL5. O caminho para o arquivo que armazena o modelo (template) é passado como o segundo parâmetro para o ChartApplyTemplate(). O arquivo do modelo (template) é procurado de acordo com as seguintes regras:

- se a barra invertida "\" separador (escrito como "\\") é colocada no início do percurso, o modelo (template) é procurado em relação ao caminho _terminal_dados_diretório\ MQL5 (path _terminal_data_directory\MQL5);
- se não existir uma barra invertida, o modelo é procurado em relação ao arquivo executável EX5, em que é chamado o ChartApplyTemplate();
- se um modelo não é encontrado nas duas primeiras variantes, a pesquisa é executada na pasta terminal_diretorio\Profiles\Templates\.

Aqui o terminal_directory é a pasta onde o Terminal do Cliente MetaTrader 5 está sendo executado, e o terminal_dados_diretório (terminal_data_directory) é a pasta em que os arquivos editáveis são armazenados, a sua localização depende do sistema operacional,nome do usuário e as configurações de segurança do computador. Normalmente, eles são pastas diferentes, mas em alguns casos, eles podem coincidir.

A localização das pastas terminal_data_directory e terminal_directory pode ser obtida utilizando a função TerminalInfoString().

```
//--- diretório de onde o terminal é iniciado
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Terminal diretório:",terminal_path);
//--- terminal dados diretório, em que a pasta MQL5 com os EAs e indicadores estão loc
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Terminal dados diretório:",terminal_data_path);
```

Por exemplo:

```
//--- busca de um modelo (template) no terminal_dados_diretório\MQL5\
ChartApplyTemplate(0,"\\primeiro_modelo.tpl"))
//--- busca de um modelo em diretório_de_EX5_arquivo\, em seguida a pasta terminal_dac
ChartApplyTemplate(0,"segundo_modelo.tpl"))
```

© 2000-2024, MetaQuotes Ltd.

```
//--- busca de um modelo em diretório_de_EX5_arquivo\My_templates\, em seguida a pasta
ChartApplyTemplate(0,"My_templates\\terceiro_modelo.tpl"))
```

Os modelos não são recursos, eles não podem ser incluídos em um arquivo executável EX5.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Exemplo de aplicação do modelo, localizado em \MQL5\Files
   if(FileIsExist("my_template.tpl"))
     {
      Print("O arquivo my_template.tpl foi encontrado em \Files'");
      //--- Aplicar modelo
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("O modelo 'my_template.tpl' foi aplicado com sucesso");
         //--- redesenhar gráfico
         ChartRedraw();
        }
      else
         Print("Falha ao aplicar'my_template.tpl', código de erro ",GetLastError());
     }
   else
     {
      Print("Arquivo 'my_template.tpl' não encontrado em"
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

Também Veja

Recursos

© 2000-2024, MetaQuotes Ltd.

ChartSaveTemplate

Salva as configurações atuais do gráfico em um template (modelo) com um nome especificado.

```
bool  ChartSaveTemplate(
   long          chart_id,      // ID Gráfico
   const string  filename       // Nome do arquivo para salvar o template (modelo)
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

filename

[in] O nome do arquivo para salvar o template (modelo). A extensão ".tpl" será adicionada automaticamente no arquivo, onde não é preciso especificá-lo. O template (modelo) é salvo no **terminal_directory**\Profiles\Templates\ e pode ser utilizado para a aplicação manual no terminal. Se já existir um template (modelo) com o mesmo nome de arquivo, os conteúdos deste arquivo serão substituídos.

Valor do Retorno

Se bem sucedida, a função retorna true, caso contrário ela retorna false. Para obter informação sobre o erro, chamar a função GetLastError().

Observação

Usando templates (modelos), você pode salvar as configurações de gráfico com todos os indicadores e objetos gráficos aplicados, para depois aplicá-los em outro gráfico.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- parâmetros de entrada
input string               symbol="GBPUSD";  // O símbolo do novo gráfico
input ENUM_TIMEFRAMES      period=PERIOD_H3; // O prazo do novo gráfico
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Primeiros indicadores anexados ao gráfico
   int handle;
```

```
//--- Preparar indicador para uso
   if(!PrepareZigzag(NULL,0,handle)) return; // Falhou, sair
//--- Anexar o indicador ao atual gráfico, mas em janela separada.
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Falha para anexar no gráfico %s/%s um indicador com o handle=%d. Có
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- Finalizar a operação do programa
      return;
     }
//--- Atualizar o gráfico para ver o indicator
   ChartRedraw();
//--- Encontre as duas últimas fracturas do zigzag
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("Falha para localizar duas últimas fracturas no ZigZag!");
      //--- Finalizar a operação do programa
      return;
     }
//--- Anexar agora um canal de desvio padrão
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("Falha ao criar objeto %s. Código de erro %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- O canal foi criado, definir o segundo ponto
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- Define uma dica de contexto (tooltip) para o canal
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demonstração no ajuda MQL5");
      //--- Atualizar o gráfico
      ChartRedraw();
     }
//--- Salve o resultado em um template (modelo)
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- Abrir um novo gráfico e aplicar um modelo salvo a ele
   long new_chart=ChartOpen(symbol,period);
   //--- Habilitar uma dica de contexto (tootips) para objetos gráficos
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Aplicar uma template (modelo) salvo para o gráfico
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Cria um manipulador ziguezague e garante disponibilidade de seus dados|
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- O indicador de ziguezague deve estar localizado em terminal_data_folder\MQL5\Exa
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Falha ao criar manipulador do indicador Zigzag. Código de erro
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- Quando é criado um manipulador de indicador, requer tempo para calcular valores
   int k=0; // O número de tentativas de espera para o cálculo de indicadores
//--- Aguarda até que o cálculo faça um loop (volta completa), pausando por 50 milisse
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- Mostrar o número de tentativas
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- Espera 50 milissegundos até que o indicador seja calculado
      Sleep(50);
      //--- Se mais de 100 tentativas, então algo está errado
      if(k>100)
        {
         //--- Reportar um problema
         PrintFormat("Falha ao calcular o indicador para %d tentativas!");
         //--- Terminar a operação do programa
         return false;
        }
     }
//--- Tudo está pronto, o indicador é criado e os valores são calculados
   return true;
  }
//+------------------------------------------------------------------+
//|Busca para as últimas duas fracturas de zigzag e lugar para arrays|
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // Uma array para os valores de ziguezague
   datetime times[];         // Uma array para obter tempo
   int size=100;             // Tamanho da array
```

```
   ResetLastError();
//--- Copiar os últimos 100 valores do indicador
   int copied=CopyBuffer(handle,0,0,size,values);
//--- Conferir o número de valores copiados
   if(copied<100)
     {
      PrintFormat("%s: Falha para copiar valores %d do indicador com o handle=%d. Cód
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- Definir a ordem de acesso para a array como um tempo de series (timeseries)
   ArraySetAsSeries(values,true);
//--- Escreva aqui o número de baras, em que foram encontradas fracturas
   int positions[];
//--- Definir tamanhos de array
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- Contadores
   int i=0,k=0;
//--- Comece a procurar por fracturas
   while(i<100)
     {
      double v=values[i];
      //--- Nós não estamos interessados em valores vazios
      if(v!=0.0)
        {
         //--- Relembra o número de barras
         positions[k]=i;
         //--- Relembra o valor do zigzag sobre a fractura
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- Aumentar o contador
         k++;
         //--- Se duas fraturas encontrados, quebrar o loop (ciclo)
         if(k>2) break;
        }
      i++;
     }
//--- Definir a ordem de acesso às arrays como nas séries de tempo (timeseries)
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: Falha ao copiar valores %d a partir CopyTime(). Código de erro
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- Abra o tempo da abertura da barra, que ocorreu nas duas últias fracturas
   get_times[0]=times[positions[1]];// O último, mas um valor será escrito como a prin
   get_times[1]=times[positions[2]];// O valor em terceiro lugar, a partir do final se
   PrintFormat("%s: primeiro=%s,  segundo=%s",__FUNCTION__,TimeToString(get_times[1]),
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Bem sucedido
   return true;
  }
```

Também Veja

ChartApplyTemplate(), Recursos

© 2000-2024, MetaQuotes Ltd.

ChartWindowFind

A função retorna o número para uma sub-janela quando um indicador é desenhado. Existem 2 variantes da função.

1. As pesquisas de função no gráfico indicado para a sub-janela especificado com o "nome abreviado" do indicador (o nome abreviado é exibida na parte superior esquerda da sub-janela), e retorna o número da sub-janela em caso de sucesso.

```
int  ChartWindowFind(
   long     chart_id,                  // identificar gráfico
   string   indicator_shortname        // nome abreviado do indicador, veja INDICATOR
```

2. A função deve ser chamado a partir de um indicador personalizado. Retorna o número de uma sub-janela onde um indicador está funcionando.

```
int  ChartWindowFind();
```

Parâmetros

chart_id

[in] ID Gráfico. 0 (zero) indica a sub-janela gráfico principal.

indicator_shortname

[in] Nome abreviado do indicador.

Valor do Retorno

Número da sub-janela em caso de sucesso. Em caso de falha a função retorna -1.

Observação

Se a segunda variante da função (sem parâmetros) é chamado de um script ou Expert Advisor, a função retorna -1.

Não confundir o nome abreviado de um indicador e um nome de arquivo, que é especificado quando o indicador é criado usando as funções iCustom() e IndicatorCreate(). Se o nome abreviado de um indicador não é definido de forma explicita, então o nome do arquivo contendo o código-fonte do indicador será especificado durante a compilação.

É importante para formar corretamente o nome abreviado de um indicador, que é registrado na propriedade INDICATOR_SHORTNAME usando a função IndicatorSetString(). É recomendado que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque sendo o indicador excluído do gráfico pela função ChartIndicatorDelete(), ele é identificado pelo nome abreviado.

Exemplo:

```
#property script_show_inputs
//--- parâmetros de entrada
input string   shortname="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| Retorna o múmero da janela do gráfico com este indicador         |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- a função é chamada a partir do indicador, o nome não é necessário
      window=ChartWindowFind();
     }
   else
     {
      //--- a função é chamada a partir de um Expert Advisor ou script
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Erro = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortname);
   if(window!=-1)
      Print("Indicador "+shortname+" está na janela #"+(string)window);
   else
      Print("Indicador "+shortname+" não foi encontrado. window = "+(string)window);
  }
```

Também Veja

ObjectCreate(), ObjectFind()

© 2000-2024, MetaQuotes Ltd.

ChartTimePriceToXY

Converte as coordenadas do gráfico a partir da representação tempo/preço para as coordenadas X e Y.

```
bool  ChartTimePriceToXY(
   long            chart_id,      // ID Gráfico
   int             sub_window,    // O número de sub-janelas
   datetime        time,          // Tempo do Gráfico
   double          price,         // Preço no gráfico
   int&            x,             // A coordenada X para o tempo no gráfico
   int&            y              // A coordenada Y para o tempo no gráfico
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

sub_window

[in] O numero da sub-janela do gráfico. 0 (zero) significa a principal janela do gráfico.

time

[in] O valor do tempo do gráfico, valores em pixels para o qual vai ser recebido ao longo do eixo X. A origem está no canto superior esquerdo da principal janela do gráfico.

price

[in] O valor do preço no gráfico, valores em pixels para o qual vai ser recebido ao longo do eixo Y. A origem está no canto superior esquerdo da principal janela do gráfico.

x

[out] A variável, dentro do qual a conversão do tempo para X será recebida.

y

[out] A variável, dentro do qual a conversão do preço para Y será recebida.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informação sobre o erro, chamar a função GetLastError().

Também Veja

ChartXYToTimePrice()

© 2000-2024, MetaQuotes Ltd.

ChartXYToTimePrice

Converte as coordenadas X e Y no gráfico para os valores de tempo e preço.

```
bool  ChartXYToTimePrice(
   long            chart_id,      // ID Gráfico
   int             x,             // A coordenada X no Gráfico
   int             y,             // A coordenada Y no Gráfico
   int&            sub_window,    // O numero da sub-janela
   datetime&       time,          // Tempo no gráfico
   double&         price          // Preço no gráfico
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

x

[in] A coordenada X.

y

[in] A coordenada Y.

sub_window

[out] A variável, em que o número de sub-janela gráfico será escrito. 0 (zero) significa a principal janela do gráfico.

time

[out] O valor do tempo do gráfico, valores em pixels para o qual vai ser recebido ao longo do eixo X. A origem está no canto superior esquerdo da principal janela do gráfico.

price

[out] O valor do preço no gráfico, valores em pixels para o qual vai ser recebido ao longo do eixo Y. A origem está no canto superior esquerdo da principal janela do gráfico.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informação sobre o erro, chamar a função GetLastError().

Exemplo:

```
//+------------------------------------------------------------------+
//| Função ChartEvent                                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Mostra os parâmetros do evento no gráfico
```

© 2000-2024, MetaQuotes Ltd.

```
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spa
//--- Se este é um evento de um mouse clicando no gráfico
   if(id==CHARTEVENT_CLICK)
     {
      //--- Preparar variáveis
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- Converter as coordenadas X e Y em termos de data/hora
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Janela=%d X=%d  Y=%d  =>  Tempo=%s  Preço=%G",window,x,y,TimeToS
         //--- Executar a conversão inversa: (X,Y) => (Tempo,Preço)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Tempo=%s  Preço=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,
         else
            Print("ChartTimePriceToXY código de erro de retorno: ",GetLastError());
         //--- delete lines
         ObjectDelete(0,"V Line");
         ObjectDelete(0,"H Line");
         //--- create horizontal and vertical lines of the crosshair
         ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
         ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
         ChartRedraw(0);
        }
      else
         Print("ChartXYToTimePrice código de erro de retorno: ",GetLastError());
      Print("+-----------------------------------------------------------------+");
     }
  }
```

Também Veja

ChartTimePriceToXY()

© 2000-2024, MetaQuotes Ltd.

ChartOpen

Abre um novo gráfico com o símbolo e período especificado.

```
long  ChartOpen(
   string           symbol,      // Nome Simbolo
   ENUM_TIMEFRAMES  period       // Período
   );
```

Parâmetros

symbol

[in] Simbolo do gráfico. NULL significa o símbolo do gráfico de corrente (Expert Advisor é anexado).

period

[in] Período do Gráfico (Prazo). Pode ser um dos valores de ENUM_TIMEFRAMES. 0 (zero) significa o período do atual gráfico.

Valor do Retorno

Se com sucesso, retorna à abertura do ID gráfico. Do contrário retorna 0(zero).

Observação

O número máximo possível de cartas abertas simultaneamente no terminal não pode exceder o valor CHARTS_MAX.

© 2000-2024, MetaQuotes Ltd.

ChartFirst

Retorna o ID do primeiro gráfico do terminal de cliente.

```
long  ChartFirst();
```

Valor do Retorno

ID Gráfico.

© 2000-2024, MetaQuotes Ltd.

ChartNext

Retorna o ID do gráfico do próximo gráfico para um especificado.

```
long  ChartNext(
   long  chart_id      // ID gráfico
   );
```

Parâmetros

chart_id

[in] ID Gráfico. 0 (zero) não significa o quadro atual. 0 (zero) significa "retornar o ID do primeiro gráfico".

Valor do Retorno

ID Gráfico. Se este for o fim da lista de gráfico, ele retorna -1.

Exemplo:

```
//--- variáveis para ID gráfico
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);
   while(i<limit)// Temos certamente não mais do que 100 gráficos abertos
     {
      currChart=ChartNext(prevChart); // Obter o ID do novo gráfico usando o ID gráfic
      if(currChart<0) break;          // Ter atingido o fim da lista de gráfico
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// vamos salvar o ID do gráfico atual para o ChartNext()
      i++;// Não esqueça de aumentar o contador
     }
```

© 2000-2024, MetaQuotes Ltd.

ChartClose

Fechando um gráfico especificado.

```
bool  ChartClose(
   long  chart_id=0      // ID gráfico
   );
```

Parâmetros

chart_id=0

[in] ID Gráfico. Significa o gráfico atual.

Valor do Retorno

Se com sucesso, retorna true, caso contrário é false.

© 2000-2024, MetaQuotes Ltd.

ChartSymbol

Retorna o nome do símbolo do gráfico especificado.

```
string  ChartSymbol(
   long  chart_id=0      // ID gráfico
   );
```

Parâmetros

chart_id=0

[in] ID Gráfico. Significa o gráfico atual.

Valor do Retorno

Se o gráfico não existe, o resultado será uma string (seqüência) vazia.

Também Veja

ChartSetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

ChartPeriod

Retorna o prazo doperiodo do gráfico especificado.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // ID gráfico
   );
```

Parâmetros

chart_id=0

[in] ID Gráfico. Significa o gráfico atual.

Valor do Retorno

A função retorna um valor de ENUM_TIMEFRAMES. Se gráfico não existir retorna 0 (zero).

© 2000-2024, MetaQuotes Ltd.

ChartRedraw

Chama um redesenho forçado de um gráfico especificado.

```
void  ChartRedraw(
   long  chart_id=0      // ID gráfico
   );
```

Parâmetros

chart_id=0

[in] ID Gráfico. Significa o gráfico atual.

Observação

Normalmente é utilizado após mudar a propriedade do objeto.

Também Veja

Funções de objetos

© 2000-2024, MetaQuotes Ltd.

ChartSetDouble

Define um valor para a propriedade de correspondência de um gráfico especificado. Propriedade de gráfico deve ser do tipo double. O comando é adicionado para plotar uma fila de mensagens de gráfico e será executado após o processamento de todos os comandos anteriores.

```
bool  ChartSetDouble(
   long                          chart_id,      // ID Gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // ID Propriedade
   double                        value          // Valor
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Pode ser um valores de ENUM_CHART_PROPERTY_DOUBLE (exceto apenas a leitura das propriedades).

value

[in] Valor da Propriedade.

Valor do Retorno

Retorna verdadeiro se o comando foi adicionado à fila do gráfico, caso contrário será falso. Para obter uma informação sobre o erro, chamar a função GetLastError().

Observação

A função é assíncrona, isto é, ela não espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado, em vez disso, ela retorna o controle imediatamente. A propriedade será alterada somente após o processamento do comando na fila do gráfico. Para executar imediatamente comandos na fila de agendamento, você precisa chamar a função ChartRedraw.

Se quiser alterar várias propriedades gráficas de uma só vez, você deverá executar as funções correspondentes (ChartSetString, ChartSetDouble, ChartSetString) num bloco de código e, em seguida, chamar uma vez ChartRedraw.

Para verificar o resultado da execução, você pode usar a função que solicita a propriedade especificada do gráfico (ChartGetInteger, ChartGetDouble, ChartSetString). Ao fazer isto, deve-se ter em mente que essas funções são síncronas e esperam o resultado da execução.

© 2000-2024, MetaQuotes Ltd.

ChartSetInteger

Define um valor para a propriedade de correspondência de um gráfico especificado. Propriedade do gráfico deve obrigatoriamente ser datetime, int, color, bool ou char. O comando é adicionado para plotar uma fila de mensagens de gráfico e será executado após o processamento de todos os comandos anteriores.

```
bool  ChartSetInteger(
   long                          chart_id,      // ID Gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // ID Propriedade
   long                          value          // Valor
   );
```

Define o valor para a respectiva propriedade da subjanela especificada

```
bool  ChartSetInteger(
   long                          chart_id,      // ID Gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // ID Propriedade
   int                           sub_window,    // Número da sub-janela
   long                          value          // Valor
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Pode ser um valores de ENUM_CHART_PROPERTY_INTEGER (exceto apenas a leitura das propriedades).

sub_window

[in] Número de sub-janela do gráfico. Para o primeiro caso, o valor padrão é 0 (janela do gráfico principal). A maioria das propriedades não requerem uma número de sub-janela.

value

[in] Valor de propriedade.

Valor do Retorno

Retorna verdadeiro se o comando foi adicionado à fila do gráfico, caso contrário será falso. Para obter uma informação sobre o erro, chamar a função GetLastError().

Observação

A função é assíncrona, isto é, ela não espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado, em vez disso, ela retorna o controle imediatamente. A propriedade será alterada somente após o processamento do comando na fila do gráfico. Para executar imediatamente comandos na fila de agendamento, você precisa chamar a função ChartRedraw.

Se quiser alterar várias propriedades gráficas de uma só vez, você deverá executar as funções correspondentes (ChartSetString, ChartSetDouble, ChartSetString) num bloco de código e, em seguida, chamar uma vez ChartRedraw.

© 2000-2024, MetaQuotes Ltd.

Para verificar o resultado da execução, você pode usar a função que solicita a propriedade especificada do gráfico (ChartGetInteger, ChartGetDouble, ChartSetString). Ao fazer isto, deve-se ter em mente que essas funções são síncronas e esperam o resultado da execução.

Exemplo:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- incluir mensagens sobre o movimento do mouse na janela do gráfico
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- a atualização forçada das propriedades do gráfico garante a prontidão para o pro
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartSetString

Define um valor para a propriedade de correspondência de um gráfico especificado. Propriedade de Gráfico deve ser do tipo string (seqüência). O comando é adicionado para plotar uma fila de mensagens de gráfico e será executado após o processamento de todos os comandos anteriores.

```
bool  ChartSetString(
   long                          chart_id,     // ID Gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id,      // ID Propriedade
   string                        str_value     // Valor
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Pode ser um valores de ENUM_CHART_PROPERTY_STRING (exceto apenas a leitura das propriedades).

str_value

[in] Valor de propriedade string. Comprimento da seqüência não pode exceder 2045 caracteres (caracteres extras serão truncados).

Valor do Retorno

Retorna verdadeiro se o comando foi adicionado à fila do gráfico, caso contrário será falso. Para obter uma informação sobre o erro, chamar a função GetLastError().

Observação

ChartSetString pode ser usado para um comentário de saída no gráfico ao invés da função Comment.

A função é assíncrona, isto é, ela não espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado, em vez disso, ela retorna o controle imediatamente. A propriedade será alterada somente após o processamento do comando na fila do gráfico. Para executar imediatamente comandos na fila de agendamento, você precisa chamar a função ChartRedraw.

Se quiser alterar várias propriedades gráficas de uma só vez, você deverá executar as funções correspondentes (ChartSetString, ChartSetDouble, ChartSetString) num bloco de código e, em seguida, chamar uma vez ChartRedraw.

Para verificar o resultado da execução, você pode usar a função que solicita a propriedade especificada do gráfico (ChartGetInteger, ChartGetDouble, ChartSetString). Ao fazer isto, deve-se ter em mente que essas funções são síncronas e esperam o resultado da execução.

Exemplo:

```
void OnTick()
  {
//---
   double Ask,Bid;
```

© 2000-2024, MetaQuotes Ltd.

```
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

Também Veja

Comment, ChartGetString

© 2000-2024, MetaQuotes Ltd.

ChartGetDouble

Retorna o valor de a propriedade correspondente do gráfico especificado. Propriedade do gráfico deve ser do tipo double. Existem duas variantes desta função.

1. Retorna diretamente o valor da propriedade.

```
double  ChartGetDouble(
   long                          chart_id,          // ID Gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,           // ID Propriedade
   int                           sub_window=0       // número da sub-janela, se necessá
   );
```

2. Retorna true ou false, dependendo do sucesso na execução da função. Se for bem sucedido, o valor da propriedade é colocado num alvo variável double_var passado como referência.

```
bool  ChartGetDouble(
   long                          chart_id,          // ID Gráfico
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,           // ID Propriedade
   int                           sub_window,        // Número da sub-janela
   double&                       double_var         // Alvo variável para a propriedade
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Este valor pode ser um dos valores ENUM_CHART_PROPERTY_DOUBLE.

sub_window

[in] Número de sub-janela do gráfico. Para o primeiro caso, o valor padrão é 0 (janela do gráfico principal). A maioria das propriedades não requerem uma número de sub-janela.

double_var

[out] Variável alvo de tipo double (duplo) para a propriedade solicitada.

Valor do Retorno

Valor de tipo double.

Para a segunda chamada caso retorne verdadeiro (true) se a propriedade especificada está disponível e seu valor foi armazenado dentro da variável double_var, caso contrário retorna false. Para obter uma informação adicional sobre o erro, é necessário chamar a função GetLastError().

Observação

A função é síncrona, isto é, ela espera a execução de todos os comandos, que foram colocados na fila do gráfico antes de serem chamados.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
   double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
   Print("CHART_PRICE_MIN =",priceMin);
   Print("CHART_PRICE_MAX =",priceMax);
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartGetInteger

Retorna o valor de uma propriedade correspondente do gráfico especificado. Propriedade do gráfico é obrigatório ser do tipo datetime, int ou bool . Existem duas variantes das chamadas desta função.

1. Retorna diretamente o valor da propriedade.

```
long  ChartGetInteger(
   long                          chart_id,          // ID Gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,           // ID Propriedade
   int                           sub_window=0       // número da sub-janela, se necessa
   );
```

2. Retorna true ou false, dependendo do sucesso na execução da função. Se for bem sucedido, o valor da propriedade é colocado num alvo variável long_var passado como referência.

```
bool  ChartGetInteger(
   long                          chart_id,          // ID Gráfico
   ENUM_CHART_PROPERTY_INTEGER   prop_id,           // ID Propriedade
   int                           sub_window,        // Número da sub-janela
   long&                         long_var           // Alvo variável para a propriedade
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Este valor pode ser um dos valores ENUM_CHART_PROPERTY_INTEGER.

sub_window

[in] Número de sub-janela do gráfico. Para o primeiro caso, o valor padrão é 0 (janela do gráfico principal). A maioria das propriedades não requerem uma número de sub-janela.

long_var

[out] Alvo variável do tipo long para a propriedade solicitada.

Valor do Retorno

Valor de tipo long.

Para a segunda chamada caso retorne true se a propriedade especificada está disponível e seu valor foi armazenado dentro da variável long_var, caso contrário retorna false. Para obter uma informação adicional sobre o error, é necessário chamar a função GetLastError().

Observação

A função é síncrona, isto é, ela espera a execução de todos os comandos, que foram colocados na fila do gráfico antes de serem chamados.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   Print("CHART_HEIGHT_IN_PIXELS =",height,"pixels");
   Print("CHART_WIDTH_IN_PIXELS =",width,"pixels");
  }
```

© 2000-2024, MetaQuotes Ltd.

ChartGetString

Retorna o valor de uma propriedade correspondente do gráfico especificado. Propriedade de Gráfico deve ser do tipo string. Existem duas variantes das chamadas desta função.

1. Retorna diretamente o valor da propriedade.

```
string  ChartGetString(
   long                          chart_id,          // ID Gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id            // ID Propriedade
   );
```

2. Retorna true ou false, dependendo do sucesso na execução da função. Se for bem sucedido, o valor da propriedade é colocado num alvo variável string_var passado como referência.

```
bool  ChartGetString(
   long                          chart_id,         // ID Gráfico
   ENUM_CHART_PROPERTY_STRING    prop_id,          // ID Propriedade
   string&                       string_var        // Alvo variável para a propriedade
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

prop_id

[in] ID Propriedade do gráfico. Este valor pode ser um dos valores ENUM_CHART_PROPERTY_STRING.

string_var

[out] Alvo variável do tipo string para a propriedade solicitada.

Valor do Retorno

Valor de tipo string.

Para a segunda chamada caso retorne true se a propriedade especificada está disponível e seu valor foi armazenado dentro da variável string_var, caso contrário retorna false. Para obter uma informação adicional sobre o error, é necessário chamar a função GetLastError().

Observação

ChartGetString pode ser usado para leitura dos comentários plotados no gráfico usando funções Comment ou ChartSetString.

A função é síncrona, isto é, ela espera a execução de todos os comandos, que foram colocados na fila do gráfico antes de serem chamados.

Exemplo:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
```

© 2000-2024, MetaQuotes Ltd.

```
  ChartRedraw();
  Sleep(1000);
  string comm=ChartGetString(0,CHART_COMMENT);
  Print(comm);
 }
```

Também Veja

Comment, ChartSetString

© 2000-2024, MetaQuotes Ltd.

ChartNavigate

Executa deslocamento do gráfico especificado pelo número especificado de barras em relação a posição especificada no gráfico.

```
bool  ChartNavigate(
   long                  chart_id,      // ID Gráfico
   ENUM_CHART_POSITION   position,      // Posição
   int                   shift=0        // Valor deslocamento
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

position

[in] Posição do gráfico para realizar um deslocamento. Pode ser um dos valores ENUM_CHART_POSITION.

shift=0

[in] Número de barras para deslocar o gráfico. Valor positivo significa que o deslocamento é para a direita (para o fim do gráfico), o valor negativo significa que o desvio é para a esquerda (para o início do gráfico). O deslocamento zero pode ser usado para navegar ao início ou para ao fim do gráfico.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- manipular gráfico atual
   long handle=ChartID();
   string comm="";
   if(handle>0) //se bem sucedida, configurar o gráfico
     {
      //--- desabilitar auto rolagem
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- definir uma mudança da margem direita do gráfico
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- desenhar castiçais
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- definir o modo de exibição para volume de tick (ponto)
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- preparar um texto para a saída em Comment()
```

© 2000-2024, MetaQuotes Ltd.

```
comm="Rolar 10 barras para a direita do começo do histórico";
//--- mostrar comentário
Comment(comm);
//--- rolar 10 barras para a direita do começo do histórico
ChartNavigate(handle,CHART_BEGIN,10);
//--- começar com o número da primeira barra visível no gráfico (numeração como
long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
//--- adicionar linha de caractere
comm=comm+"\r\n";
//--- adicionar comentário
comm=comm+"A primeira barra é o número no gráfico "+IntegerToString(first_bar)+"
//--- mostrar comentário
Comment(comm);
//--- aguardar 5 segundos para ver como o gráfico se move
Sleep(5000);

//--- adicionar texto de cometário
comm=comm+"\r\n"+"Rolar 10 barras para a esquerda da margem direita do gráfico";
Comment(comm);
//--- rolar 10 barras para a esquerda da margem direita do gráfico
ChartNavigate(handle,CHART_END,-10);
//--- começar com o número da primeira barra visível no gráfico (numeração como
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"A primeira barra é o número no gráfico "+IntegerToString(first_bar)+"
Comment(comm);
//--- aguardar 5 segundos para ver como o gráfico se move
Sleep(5000);

//--- novo bloco de rolagem do gráfico
comm=comm+"\r\n"+"Rolar 300 barras para a direita do início do histórico";
Comment(comm);
//--- rolar 300 barras para a direita do início do histórico
ChartNavigate(handle,CHART_BEGIN,300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"A primeira barra é o número no gráfico "+IntegerToString(first_bar)+"
Comment(comm);
//--- aguardar 5 segundos para ver como o gráfico se move
Sleep(5000);

//--- novo bloco de rolagem do gráfico
comm=comm+"\r\n"+"Rolar 300 barras para a esquerda da margem direita do gráfico"
Comment(comm);
//--- rolar 300 barras à esquerda da margem direita do gráfico
ChartNavigate(handle,CHART_END,-300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"A primeira barra é o número no gráfico "+IntegerToString(first_bar)+"
```

© 2000-2024, MetaQuotes Ltd.

```
        Comment(comm);
      }
   }
```

© 2000-2024, MetaQuotes Ltd.

ChartID

Retorna o ID do gráfico atual.

```
long  ChartID();
```

Valor do Retorno

Valor tipo long.

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorAdd

Adiciona um indicador com o identificador especificado para uma janela de gráfico especificado. Indicador e gráfico devem ser gerados sobre o mesmo símbolo e prazo.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // ID Gráfico
   int   sub_window                 // número da sub-janela
   int   indicator_handle           // manipulador do indicador
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

sub_window

[in] O número de sub-janelas do gráfico. 0 (zero) significa a principal janela do gráfico. Para adicionar um indicador em uma nova janela,o parâmetro obrigatoriamente deve ser maior do que o índice da última janela existente, isto é, igual a CHART_WINDOWS_TOTAL. Se o valor do parâmetro é maior que CHART_WINDOWS_TOTAL, uma nova janela não será criada, e o indicador não será adicionado.

indicator_handle

[in] O identificador do indicador.

Valor do Retorno

A função retorna verdadeira (true) em caso de sucesso, caso contrário ela retorna false. A fim de obter informação sobre _erro, chama a função GetLastError(). Erro 4114 significa que o gráfico e um indicador adicionado difere pelo seus símbolos ou prazo.

Observação

Se um indicador deve ser elaborado numa sub-janela separada (por exemplo, construindo um iMACD ou um indicador personalizado com propriedade especificada #property indicator_separate_window) é aplicada à janela de gráfico principal,não pode ser visível embora ainda estará presente na lista de indicadores. Isto significa que a escala do indicador é diferente da escala do gráfico de preço, e aplicado ao valor do indicador não se enquadra na faixa visualizada do gráfico de preço. Neste caso, GetLastError() retorna código zero indicando a ausência de um erro. Os valores de tal indicador "invisível" pode ser visto na Janela de Dados (Data Window) e recebidos a partir de outras aplicações MQL5.

Exemplo:

```
#property description "Expert Advisor demonstrando o trabalho com a função ChartIndica
#property description "Após o lançamento no gráfico (e receber o erro no Jornal), abri
#property description "Propriedades do Expert Advisor e especificar <símbolo> correto
#property description "Indicador MACD sera adicionado no grafico."

//--- parâmetros de entrada
input string          symbol="AUDUSD";     // nome do símbolo
input ENUM_TIMEFRAMES period=PERIOD_M12;   // time frame
```

© 2000-2024, MetaQuotes Ltd.

```
input int    fast_ema_period=12;          // período rápido MACD
input int    slow_ema_period=26;          // período lento MACD
input int      signal_period=9;           // período de sinal
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // tipo de preço para cálculo do iMACD

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- tentar adicionar o indicador no gráfico
   if(!AddIndicator())
     {
      //--- Função AddIndicator() recusou acrescentar o indicador no gráfico
      int answer=MessageBox("Você quer adicionar MACD do gráfico de qualquer maneira?"
                            "Símbolo e/ou prazo de tempo incorreto para adicionar o in
                            MB_YESNO // "Sim" e "Não" botões de seleção será mostrado
                            );
      //--- se um usuário ainda insiste no uso incorreto de ChartIndicatorAdd()
      if(answer==IDYES)
        {
         //--- em primeiro lugar, uma entrada jornal será publicado sobre isso
         PrintFormat("Atenção! %s: Tentando adicionar indicador MACD(%s/%s) em gráfico
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //---receber o número de uma nova sub-janela, para a qual vamos tentar adicio
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- agora faz uma tentativa fadada a falhar
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("Falha ao adicionar o indicador MACD %d na janela do gráfico.
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
// Assessor especialista (expert advisor) não executa nada
  }
//+------------------------------------------------------------------+
//| Função para verificar e adicionar o indicador no gráfico         |
//+------------------------------------------------------------------+
bool AddIndicator()
```

```
  {
//--- mensagem exibida
   string message;
//--- verificar se o símbolo do indicador e símbolo do gráfico combinam entre si
   if(symbol!=_Symbol)
     {
      message="Exibindo o uso da função Demo_ChartIndicatorAdd():";
      message=message+"\r\n";
      message=message+"Não é possível adicionar o indicador calculado com outro símbol
      message=message+"\r\n";
      message=message+"Especifique o símbolo gráfico na propriedade do Expert Advisor
      Alert(message);
      //---saída prematura, o indicador não será adicionado no gráfico
      return false;
     }
//--- verificar se o indicador e prazo do gráfico combinam entre si
   if(period!=_Period)
     {
      message="Não é possível adicionar o indicador calculado em outro prazo de tempo
      message=message+"\r\n";
      message=message+"Especifique o prazo de tempo do gráfico na propriedades do Expe
      Alert(message);
      //---saída prematura, o indicador não será adicionado no gráfico
      return false;
     }
//--- todas verificações completadas, simbolo e prazo do indicador corresponde ao grá
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  Criando indicador MACD");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_per
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("Falha para criar indicador MACD. Código de erro ",GetLastError());
        }
     }
//--- redefine o código de erro
   ResetLastError();
//--- aplicar o indicador para o gráfico
   Print(__FUNCTION__,"  Adicionando indicador MACD no gráfico");
   Print("MACD é gerada em ",symbol,"/",EnumToString(period));
//---receber o número de uma nova sub-janela, para o qual é adicionado o indicador MAC
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("Adicionado indicador MACD na janela do gráfico %d",subwindow);
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("Falha para adicionar indicador MACD na janela do gráfico %d. Código
                  subwindow,GetLastError());
     }
//--- indicador adicionado com sucesso
```

© 2000-2024, MetaQuotes Ltd.

```
    return(true);
  }
```

Veja também

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorDelete

Remove um indicador com um nome especificado a partir de janela do gráfico especificado.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // ID Gráfico
   int            sub_window             // número da sub-janela
   const string   indicator_shortname    // nome abreviado do indicador
   );
```

Parâmetros

chart_id

[in] ID Gráfico. 0 (zero) indica a sub-janela gráfico principal.

sub_window

[in] Número de sub-janela do gráfico. 0 (zero) indica o sub-janela gráfico principal.

const indicator_shortname

[in] O nome abreviado do indicador que é definido na propriedade do INDICATOR_SHORTNAME com a função IndicatorSetString(). Para obter o nome abreviado de um indicador usar a função ChartIndicatorName().

Valor do Retorno

Retorna true em caso de sucesso deletando o indicador. Do contrário ele retorna false. Para obter detalhes do erro use a função GetLastError().

Observação

Se houver dois indicadores com nove abreviados iguais na subjanela do gráfico, o primeiro da linha será deletado.

Se outros indicadores neste gráfico são baseadas nos valores do indicador que está sendo deletado, estes indicadores também serão deletados.

Não confundir o nome abreviado do indicador e o nome do arquivo que é especificado ao criar um indicador usando as funções iCustom() e IndicatorCreate(). Se o nome abreviado de um indicador não é definido de forma explicita, o nome do arquivo contendo o código-fonte do indicador será especificado durante a compilação.

Deletando um indicador de um gráfico não significa que seu cálculo será excluído da me mória do terminal. Para liberar o manuseio do indicador usar a função IndicatorRelease().

Nome abreviado do indicador deve ser formado corretamente. a propriedade INDICATOR_SHORTNAME será escrita usando a funçãoIndicatorSetString(). Sé recomendado que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque sendo o indicador excluído do gráfico pela função ChartIndicatorDelete(), ele é identificado pelo nome abreviado.

Exemplo de exclusão de um indicador após inicialização ter falhado:

```
//+------------------------------------------------------------------+
//|                                   Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar Histograma
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- buffers do indicador
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//---Vincular a array HistogramBuffer para a buffer (memória temporária) do indicador
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Construir um nome abreviado de indicador baseado nos parâmetro de entrada
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- Se a conclusão forçada de um indicador está definida, retornar um valor não-zero
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
                const long &volume[],
                const int &spread[])
  {
//--- Posição de partida para trabalhar num loop
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- Preencher o buffer do indicador com valores
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Um processador do evento de desinicializada (Deinit)             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Código de razão de finalização=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("Um indicador com um nome abreviado %s (arquivo %s) se exclui do gra
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- Analisar o resultado de chamada de ChartIndicatorDelete()
      if(!res)
        {
         PrintFormat("Falha para deletar o indicador %s partir da janela #%d. Código d
                     shortname,window,GetLastError());
        }
     }
  }
```

Também Veja

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorGet

Retorna o manuseio do indicador com nome abreviado especificado na janela do gráfico especificado.

```
int  ChartIndicatorGet(
   long           chart_id,              // ID gráfico\
   int            sub_window             // O numero da sub-janela
   const string   indicator_shortname    // Nome abreviado do indicador
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

sub_window

[in] O numero da sub-janela do gráfico. 0 (zero) significa a principal janela do gráfico.

const indicator_shortname

[in] O nome abreviado do indicador que é definido na propriedade INDICATOR_SHORTNAME será escrita usando a função IndicatorSetString(). Para obter o nome abreviado de um indicador usar a função ChartIndicatorName().

Valor do Retorno

A função retorna bem sucedida, caso contrário retorna INVALID_HANDLE. Para obter informações sobre o erro, chame a função GetLastError().

Observação

O handle do indicador obtido com ajuda da função ChartIndicatorGet() aumenta o contador interno de uso desse indicador. O sistema de execução do terminal mantém carregados todos os indicadores cujo contador é maior que zero. Por isso, um handle de indicador desnecessário deve ser imediata e explicitamente liberado com ajuda de IndicatorRelease() no mesmo programa que o recebe, conforme mostrado no exemplo abaixo. Caso contrário, será impossível encontrar o identificador "abandonado" e liberá-lo corretamente desde outro programa.

Ao criar um indicador, tenha cuidado para formar sua abreviação, o que está escrito na propriedade INDICATOR_SHORTNAME usando a função IndicatorSetString(). É recomendado que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque sendo o indicador excluído do gráfico pela função ChartIndicatorGet() ele é identificado pelo nome abreviado.

Outra forma de identificar o indicador é obtendo uma lista de seus parâmetros para um manipulador, usando a função IndicatorParameters() e, em seguida, analisar os valores obtidos.

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- O número de janelas no gráfico (ao menos uma janela principal está sempre pre
```

© 2000-2024, MetaQuotes Ltd.

```
    int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
    //--- Verifique todas as janelas
    for(int w=0;w<windows;w++)
      {
       //--- o número de indicadores nesta janela/sub-janela
       int total=ChartIndicatorsTotal(0,w);
       //--- Passar por todos os indicadores na janela
       for(int i=0;i<total;i++)
         {
          //--- obtém o nome abreviado do indicador
          string name=ChartIndicatorName(0,w,i);
          //--- obter o manipulador (handle) de um indicador
          int handle=ChartIndicatorGet(0,w,name);
          //--- Adicionar o log
          PrintFormat("Janela=%d,  índice=%d,  nome=%s,  manipulador (handle)=%d",w,i,n
          //--- You should obligatorily release the indicator handle when it is no long
          IndicatorRelease(handle);
         }
      }
  }
```

Também Veja

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorName

Retorna o nome abreviado do indicador pelo número na lista de indicadores sobre a janela do gráfico especificado.

```
string  ChartIndicatorName(
   long  chart_id,       // ID gráfico
   int   sub_window      // numero de sub-janelas
   int   index           // índice do indicador na lista de indicadores adicionada a su
   );
```

Parâmetros

chart_id

[in] ID Gráfico. 0 (zero) indica a sub-janela gráfico principal.

sub_window

[in] Número de sub-janela do gráfico. 0 (zero) indica o sub-janela gráfico principal.

index

[in] índice do indicador na lista de indicadores. A numeração de indicadores começam com zero, ou seja, o primeiro indicador da lista tem o índice 0. Para obter o código do último erro, você também pode usar a função ChartIndicatorsTotal().

Valor do Retorno

O nome abreviado do indicador que é definido na propriedade INDICATOR_SHORTNAME com a função IndicatorSetString(). Para obter detalhes do erro use a função GetLastError().

Observação

Não confundir o nome abreviado do indicador e o nome do arquivo que é especificado ao criar um indicador usando as funções iCustom() e IndicatorCreate(). Se o nome abreviado de um indicador não é definido de forma explicita, o nome do arquivo contendo o código-fonte do indicador será especificado durante a compilação.

Nome abreviado do indicador deve ser formado corretamente. a propriedade INDICATOR_SHORTNAME será escrita usando a funçãoIndicatorSetString(). Sé recomendado que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque sendo o indicador excluído do gráfico pela função ChartIndicatorDelete(), ele é identificado pelo nome abreviado.

Também Veja

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartIndicatorsTotal

Retorna o número de todos indicadores utilizados para a janela do gráfico especificado

```
int  ChartIndicatorsTotal(
   long  chart_id,       // ID gráfico
   int   sub_window      // numero de sub-janelas
   );
```

Parâmetros

chart_id

[in] ID Gráfico. 0 (zero) indica a sub-janela gráfico principal.

sub_window

[in] Número de sub-janela do gráfico. 0 (zero) indica o sub-janela gráfico principal.

Valor do Retorno

O número de indicadores na janela de gráfico especificado. Para obter detalhes do erro use a função GetLastError().

Observação

A função permite uma buscar através de todos os indicadores anexados ao gráfico. A função permite uma pesquisa através de todos os indicadores anexados ao gráfico. O número de todas as janelas do gráfico pode ser obtido a partir da propriedade CHART_WINDOWS_TOTAL usando a função ChartGetInteger().

Também Veja

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2024, MetaQuotes Ltd.

ChartWindowOnDropped

Retorna o número (índice) da sub-janela do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado. 0 (zero) significa a principal janela do gráfico.

```
int  ChartWindowOnDropped();
```

Valor do Retorno

Valor do tipo int.

Exemplo:

```
int myWindow=ChartWindowOnDropped();
int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("Script é executado na janela #"+myWindow+
      ". Total de janelas no gráfico "+ChartSymbol()+":",windowsTotal);
```

Também Veja

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartPriceOnDropped

Retorna a coordenada de preço do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado.

```
double  ChartPriceOnDropped();
```

Valor do Retorno

Value of double type.

Exemplo:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Também Veja

ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartTimeOnDropped

Retorna a coordenada de tempo correspondente ao ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado.

```
datetime  ChartTimeOnDropped();
```

Valor do Retorno

Valor tipo datetime.

Exemplo:

```
   datetime t=ChartTimeOnDropped();
   Print("Script foi lançado sobre "+t);
```

Também Veja

ChartXOnDropped, ChartYOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartXOnDropped

Retorna a coordenada X do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado.

```
int  ChartXOnDropped();
```

Valor do Retorno

O valor da coordenada X.

Observação

Direção do eixo X a partir da esquerda para a direita.

Exemplo:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Também Veja

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartYOnDropped

Retorna a coordenada X do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado.

```
int  ChartYOnDropped();
```

Valor do Retorno

O valor da coordenada Y.

Observação

Direção do eixo Y a partir de cima para baixo.

Também Veja

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2024, MetaQuotes Ltd.

ChartSetSymbolPeriod

Modifica o valor do símbolo e um período do gráfico especificado. A função é assíncrona, ou seja, envia o comando e não espera a conclusão da execução. O comando é adicionado para plotar uma fila de mensagens de gráfico e será executado após o processamento de todos os comandos anteriores.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // ID Gráfico
   string           symbol,        // Nome símbolo
   ENUM_TIMEFRAMES  period         // Período
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

symbol

[in] Simbolo do gráfico. Valor NULL significa o símbolo gráfico atual (Expert Advisor é anexado)

period

[in] Período do Gráfico (Prazo). Pode ser um dos valores de ENUM_TIMEFRAMES. 0 (zero) significa o período do atual gráfico.

Valor do Retorno

Retorna verdadeiro se o comando foi adicionado à fila do gráfico, caso contrário será falso. Para obter uma informação sobre o erro, chamar a função GetLastError().

Observação

O símbolo/período de mudança conduz à ré-inicialização do Expert Advisor, anexado a um gráfico.

A chamada da função ChartSetSymbolPeriod, com o mesmo símbolo e timeframe, pode ser utilizada para atualizar o gráfico (semelhante ao comando Refresh no terminal). A atualização do gráfico, por sua vez , inicia o recálculo dos indicadores ligados a ele. Assim, você pode calcular o indicador no gráfico, mesmo na ausência de ticks (por exemplo, nos fins-de-semana e feriados).

Veja Também

ChartSymbol, ChartPeriod

© 2000-2024, MetaQuotes Ltd.

ChartScreenShot

Providencia uma imagem do gráfico de seu estado atual para um formato GIF, PNG ou BMP dependendo da extensão especificada.

```
bool  ChartScreenShot(
   long            chart_id,                  // ID Gráfico
   string          filename,                  // Nome do Símbolo
   int             width,                     // Largura
   int             height,                    // Altura
   ENUM_ALIGN_MODE align_mode=ALIGN_RIGHT     // Tipo de alinhamento
   );
```

Parâmetros

chart_id

[in] ID Gráfico. Significa o gráfico atual.

filename

[in] Nome do arquivo de imagem. Não pode exceder 63 caracteres. Arquivos de captura de tela são colocados no \Files diretório.

width

[in] Captura de tela com largura em pixels.

height

[in] Captura de tela com altura em pixels.

align_mode=ALIGN_RIGHT

[in] Modo de saída de uma Captura de tela estreita. O valor da enumeração ENUM_ALIGN_MODE. ALIGN_RIGHT significa alinhamento para a margem direita (a saída a partir do final). ALIGN_LEFT significa justificar esquerda.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Observação

Se você precisa tomar uma captura de tela de um gráfico a partir de uma determinada posição, em primeiro lugar, é necessário posicionar o gráfico usando a função ChartNavigate(). Se o tamanho horizontal da captura de tela é menor do que a janela do gráfico, ou a parte direita da janela do gráfico, ou a sua parte esquerda está de saída, depende das configurações do align_mode.

Exemplo:

```
#property description "O Expert Advisor demonstra como criar uma série de captura de t
#property description  gráfico usando a função ChartScreenShot(). Por conveniência, o
#property description "mostrando sobre o gráfico. A altura e a largura das imagens são

#define        WIDTH  800     // ChartScreenShot() para Chamar a largura da imagem
#define        HEIGHT 600     // ChartScreenShot() para Chamar a altura da imagem
```

© 2000-2024, MetaQuotes Ltd.

```
//--- parâmetros de entrada
input int      pictures=5;     // O número de imagens em série
int            mode=-1;        // -1 denota uma mudança para a borda direita do gráfico
int            bars_shift=300;// O número de barras de rolagem quando o gráfico está u
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Desabilita gráfico autoscroll
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- Definir o deslocamento da borda direita do gráfico
   ChartSetInteger(0,CHART_SHIFT,true);
//--- Mostra uma castiçal (vela ou barra) do gráfico
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("Preparação do Expert Advisor está completada");
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Função ChartEvent                                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Mostra o nome da função, chamar tempo e identificador de evento
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- Manipular o evento CHARTEVENT_CLICK ("Um clique do mouse sobre o gráfico")
   if(id==CHARTEVENT_CLICK)
     {
      //--- Deslocamento inicial para a borda do gráfico
      int pos=0;
      //--- Operação com a borda esquerda do gráfico
      if(mode>0)
        {
         //--- Rolar o gráfico para a borda esquerda
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- Preparar o texto para mostrar no gráfico e um nome de arquivo
```

```
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- Mostrar o nome no gráfico como um comentário
            Comment(name);
            //--- Salvar a captura de tela do gráfico em um arquivo no terminal_direct
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Nós salvamos a captura de tela ",name);
            //---
            pos+=bars_shift;
            //--- Dê tempo ao utilizador para olhar a nova parte nova do gráfico
            Sleep(3000);
            //--- Rolar o gráfico diante da atual posição do bars_shift bars para a di
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- Altere o modo para o oposto
         mode*=-1;
        }
      else // Operação com a borda direita do gráfico
        {
         //--- Rolar o gráfico para a borda direita
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- Preparar o texto para mostrar no gráfico e um nome de arquivo
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- Mostrar o nome no gráfico como um comentário
            Comment(name);
            //--- Salvar a captura de tela do gráfico em um arquivo no terminal_direct
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Nós salvamos a captura de tela ",name);
            //---
            pos+=bars_shift;
            //--- Dê tempo ao utilizador para olhar a nova parte nova do gráfico
            Sleep(3000);
            //--- Rolar o gráfico diante da atual posição do bars_shift bars para a di
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- Altere o modo para o oposto
         mode*=-1;
        }
     }  // End of CHARTEVENT_CLICK event handling
//--- Fim do manipulador (handler) OnChartEvent()
  }
```

Também Veja

ChartNavigate(), Recursos

© 2000-2024, MetaQuotes Ltd.

Funções de Negociação (Trade)

Este é o grupo de funções destina-se à gestão de atividades de negociação.

Antes de começar a estudar as funções da plataforma, você deve compreender claramente todos os termos e funções: ordem, transação e posição:

- Ordem — solicitação feita pela empresa de corretagem para comprar ou vender um instrumento financeiro. Existem dois tipos de ordens básicos: de mercado e pendentes. Além disso, existem ordens especiais Take Profit e Stop Loss.
- Transação — fato da compra ou venda de um instrumento financeiro. A compra (Buy) acontece segundo o preço de demanda (Ask) e a venda acontece segundo o preço de oferta (Bid). A transação pode ser feita como resultado da execução de uma ordem de mercado ou ativação de uma operação pendente. Você deve ter em conta que, em alguns casos, várias transações podem ser o resultado da execução de uma ordem.
- Posição — obrigação de mercado, número de contratos comprados ou vendidos de um instrumento financeiro. Uma posição longa (Long) é um instrumento financeiro adquirido com base no aumento do preço e uma posição curta (Short) é uma obrigação colocada com base na diminuição de preços no futuro.

Informações gerais sobre operações de negociação estão disponíveis no manual do usuário do terminal do cliente.

As funções de negociação podem ser usadas em Experts Advisors e scripts. Funções de negociação só podem ser chamadas se as propriedades do Expert Advisor ou script estiverem habilitadas na sua caixa de controle "Permitir negociação ao vivo".

A negociação pode ser permitida ou proibida dependendo de vários fatores descritos na seção Trade Permission.

Função	Ação
OrderCalcMargin	Calcula a margem necessária para o tipo de ordem especificado, na moeda de depósito
OrderCalcProfit	Calcula o lucro com base nos parâmetros passados, na moeda de depósito
OrderCheck	Verifica se há fundos suficientes para executar a operação de negócio necessária.
OrderSend	Envia pedidos de negócios para um servidor
OrderSendAsync	De forma assíncrona envia pedidos de comércio sem esperar a resposta da negociação do servidor de negócio
PositionsTotal	Retorna o número de posições em aberto
PositionGetSymbol	Retorna o símbolo correspondente à posição aberta
PositionSelect	Escolhe uma posição aberta para posteriormente trabalhar com ele
PositionSelectByTicket	Selects a position to work with by the ticket number specified in it
PositionGetDouble	Retorna a propriedade solicitado de uma posição aberta (double)

© 2000-2024, MetaQuotes Ltd.

Função	Ação
PositionGetInteger	Retorna a propriedade solicitada uma posição aberta (datetime ou int)
PositionGetString	Retorna a propriedade solicitada uma posição aberta (string)
PositionGetTicket	Returns the ticket of the position with the specified index in the list of open positions
OrdersTotal	Retorna o número de ordens
OrderGetTicket	Devolver o bilhete de uma ordem correspondente
OrderSelect	Seleciona uma ordem para posteriormente trabalhar com ele
OrderGetDouble	Retorna a propriedade solicitada da ordem (double)
OrderGetInteger	Retorna a propriedade solicitada da ordem (datetime ou int)
OrderGetString	Retorna a propriedade solicitada da ordem (string)
HistorySelect	Recupera o histórico de transações e ordens para o período do tempo de servidor especificado
HistorySelectByPosition	Pedido de histórico de acordo com um identificador de posição especificada.
HistoryOrderSelect	Seleciona uma ordem no histórico para posteriormente trabalhar com ele
HistoryOrdersTotal	Retorna o número de pedidos no histórico
HistoryOrderGetTicket	Retornar bilhete ordem de uma ordem correspondente no histórico
HistoryOrderGetDouble	Retorna a propriedade solicitada de uma ordem no histórico (double)
HistoryOrderGetInteger	Retorna a propriedade solicitada de uma ordem no histórico (datetime ou int)
HistoryOrderGetString	Retorna a propriedade solicitada de uma ordem no histórico (string)
HistoryDealSelect	Seleciona um negócio no histórico para posterior chamada através de funções apropriadas
HistoryDealsTotal	Retorna o número de ofertas no histórico
HistoryDealGetTicket	Devolve um bilhete de uma quantidade correspondente no histórico
HistoryDealGetDouble	Retorna a propriedade solicitada de um acordo no histórico (double)
HistoryDealGetInteger	Retorna a propriedade solicitada de um acordo no histórico (datetime ou int)
HistoryDealGetString	Retorna a propriedade solicitada de uma operação no histórico (string)

© 2000-2024, MetaQuotes Ltd.

OrderCalcMargin

A função calcula a margem necessária para o tipo de ordem especificado, na conta corrente, no ambiente de mercado atual não tendo em conta os pedidos pendentes atuais e posições abertas. Ele permite a avaliação da margem para a operação comercial prevista. O valor é retornado na moeda da conta.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,          // tipo de ordem
   string                symbol,          // nome do símbolo
   double                volume,          // volume
   double                price,           // preço aberto
   double&               margin           // valor para a obtenção do valor da margem
   );
```

Parâmetros

action

[in] O tipo de pedido, pode ser um dos valores da enumeração ENUM_ORDER_TYPE.

symbol

[in] Nome do ativo.

volume

[in] Volume das operações de Negócios (comércio).

price

[in] Preço de abertura.

margin

[out] A variável, para o qual o valor da margem requerida será escrito no caso da função ser executada com êxito. O cálculo é realizado como se não houvesse ordens pendentes e posições abertas na conta corrente. O valor da margem depende de muitos fatores, e pode variar em diferentes ambientes de mercado.

Valor do Retorno

A função retorna true em caso de sucesso, caso contrário ela retorna false. A fim de obter informações sobre o erro, chamar a função GetLastError().

Também Veja

OrderSend(), Order Properties, Tipos de Operações de Negociação (Trade)

© 2000-2024, MetaQuotes Ltd.

OrderCalcProfit

A função calcula o lucro para a conta corrente, nas atuais condições de mercado, baseado nos parâmetros passados. A função é usado para a pré-avaliação do resultado de uma operação de negócio (comércio). O valor é retornado na moeda da conta.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,          // tipo da ordem (ORDER_TYPE_BUY ou ORDER_T
   string                symbol,          // nome do símbolo
   double                volume,          // volume
   double                price_open,      // preço de abertura
   double                price_close,     // preço de fechamento
   double&               profit           // variável para a obtenção do valor de luc
   );
```

Parâmetros

action

[in] Tipo de ordem, pode um dos dois valores da enumeração ENUM_ORDER_TYPE: ORDER_TYPE_BUY ou ORDER_TYPE_SELL.

symbol

[in] Nome do ativo.

volume

[in] Volume das operações de Negócios (comércio).

price_open

[in] Preço de abertura.

price_close

[in] Preço de fechamento.

profit

[out] A variável, para o qual o valor calculado do lucro será escrito, em caso da função ser executada com sucesso. O valor do lucro estimado depende de muitos fatores, e pode variar em diferentes ambientes de mercado.

Valor do Retorno

A função retorna true em caso de sucesso, caso contrário ela retorna false. Se um tipo de ordem inválida for especificada, a função retornará falsa. Na ordem para obter informações sobre o erro, chamar GetLastError().

Também Veja

OrderSend(), Order Properties, Tipos de Operações de Negociação (Trade)

© 2000-2024, MetaQuotes Ltd.

OrderCheck

A função OrderCheck() verifica se há dinheiro suficiente para executar uma operação de comércio necessária. Os resultados da verificação são colocadas para os campos da estrutura MqlTradeCheckResult.

```
bool  OrderCheck(
   MqlTradeRequest&        request,      // estrutura solicitada
   MqlTradeCheckResult&    result        // resultado da estrutura
   );
```

Parâmetros

request

[in] Ponteiro para a estrutura do tipo MqlTradeRequest que descreve a ação de comércio requerida.

result

[in,out] Ponteiro para a estrutura do tipo MqlTradeCheckResult, em que o resultado da verificação será colocada.

Valor do Retorno

Se os fundos não são suficientes para a operação, ou os parâmetros são preenchidos incorretamente, a função retorna falsa. Em caso de sucesso na verificação básica de estruturas (verifique os ponteiros), ele retorna verdadeira. **No entanto, isto não é uma indicação de que a operação comercial solicitada, deixe de ser executado com sucesso**. Para uma descrição mais detalhada do resultado da execução da função, analise os campos do *resultado* da estrutura.

A fim de obter informações sobre o erro, chamar a função GetLastError().

Também Veja

OrderSend(), Tipos de operações de negociação, Pedido de Estrutura de Comércio, Estrutura do Pedido pelos resultados, Estrutura de um Pedido do resultado de negociação

© 2000-2024, MetaQuotes Ltd.

OrderSend

A função OrderSend() é usada para a execução de operações de negociação enviando solicitações para um servidor de negociação.

```
bool  OrderSend(
   MqlTradeRequest&  request,      // estrutura de pedido
   MqlTradeResult&   result        // estrutura da resposta
   );
```

Parâmetros

request

[in] Ponteiro para uma estrutura de tipo MqlTradeRequest descrevendo a atividade de negociação do cliente.

result

[in,out] Ponteiro para uma estrutura do tipo MqlTradeResult descrevendo o resultado da operação comercial, no caso de uma realização bem sucedida (se True é retornado).

Valor do Retorno

Em caso de sucesso na verificação básica de estruturas (verificação do índice) retorna True. **No entanto, este não é um sinal de sucesso com execução de uma operação de negociação** . Para uma descrição mais detalhada do resultado da execução da função, analisar os domínios da estrutura de *resultado*.

Observação

Os pedidos de negociação passar por várias etapas de verificação de um servidor de negociação. Primeiro de tudo, ele verifica se todos os campos obrigatórios dos parâmetros de *solicitação* estão preenchidos corretamente. Se não houver erros nesta guia, você pode iniciar o programa no terminal cliente. Se o pedido for aceito com sucesso pelo servidor de negociação, o OrderSend() retorna True.

Recomenda-se verificar o pedido antes de enviá-lo para um servidor de negociação. Para verificar os pedidos, use a função OrderCheck(). Ele verifica se há fundos suficientes para executar a operação de negociação, e retorna muitos parâmetros úteis nos resultados da verificação do pedido de negociação:

- código de retorno contém informações sobre erros no pedido verificado;
- Valor de saldo após a execução da operação de negociação;
- valor de eqüidade que irá aparecer após a operação comercial é executado;
- valor de ponto flutuante que irá aparecer após a operação de negociação for executado;
- Margem requerida para a operação de negociação;
- quantidade de capital livre que permanecerá após a execução da operação de negociação;
- o nível da margem que será definido após a operação de negociação for executado;
- Comentário sobre o código da resposta, descrição do erro.

Ao enviar uma ordem a mercado (MqlTradeRequest.action=TRADE_ACTION_DEAL), o resultado bem-sucedido da função OrderSend() não significa que a ordem tenha sido executada (executadas as transações correspondentes): neste caso, true apenas significa que a ordem foi colocada corretamente no sistema de negociação para execução posterior. Na estrutura do resultado *result* a

© 2000-2024, MetaQuotes Ltd.

ser retornada, o servidor de negociação pode preencher os valores dos campos *deal* ou *order*, se esse dados forem conhecidos por ele, no momento da formação da resposta à chamada da OrderSend(). Em geral, o(s) evento(s) da execução das transações, da ordem correspondente, podem ocorrer após ter sido enviada a resposta à chamada da OrderSend(). Por isso, para qualquer tipo de solicitação de negociação, após obtido o resultado da execução da OrderSend(), é necessário, antes de tudo, verificar o código de retorno do servidor de negociação *retcode* e o código de resposta do sistema de negociação externo *retcode_external* (se necessário), que estão disponíveis na estrutura do resultado *result* a ser retornada.

Cada pedido aceito é armazenado no servidor de negociação esperando o processamento até que ocorre uma das condições para a sua execução:

- expiração,
- aparecimento de uma solicitação oposta,
- execução da ordem, quando o preço de execução aparece,
- uma solicitação para cancelar a ordem é recebido.

No momento do processamento da ordem, o servidor de negociações envia ao terminal uma mensagem sobre a ocorrência do evento Trade (evento de negociação), que pode ser processado pela função OnTrade().

O resultado da execução do pedido de negociação sobre o servidor enviado pela função OrderSend() pode ser controlada pelo handler (Manipulador) OnTradeTransaction. Deve ser observado que o handler (manipulador) OnTradeTransaction será chamado várias vezes durante a execução de um pedido de negociação.

Por exemplo, ao enviar uma ordem de compra de mercado, ela é tratada, uma ordem de compra apropriada é criada para a conta, a ordem é então executada e removida da lista de ordens em aberto, e então ela é adicionada ao histórico de ordens, uma apropriada operação (deal) é adicionada ao histórico e uma nova posição é criada. Função OnTradeTransaction será chamado para cada um desses eventos.

Exemplo:

```
//--- valor para ORDER_MAGIC
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Ter a certeza que é uma conta demonstração
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Operação de script não é permitido em uma conta real!");
      return;
     }
//--- Colocar ou deletar uma ordem
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- Não ordens atuais - colocar uma ordem
      uint res=SendRandomPendingOrder(order_magic);
      Print("Retorna código do servidor de negociação",res);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
    else // existem ordens - deletar ordens
      {
       DeleteAllOrdersByMagic(order_magic);
      }
//---
  }
//+------------------------------------------------------------------+
//| Receber o número das atuais ordens atuais com especificação      |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- passar por todas as ordens pendentes
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| Deletar todas as ordens pendentes com especificação ORDER_MAGIC  |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- passar por todas as ordens pendentes
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- ordem com apropriado ORDER_MAGIC
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- escrever a resposta do servidor para log
            Print(__FUNCTION__,": ",result.comment," reply code ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------+
//| Definir uma ordem pendente de maneira aleatória                  |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- preparar um pedido
```

© 2000-2024, MetaQuotes Ltd.

```
   MqlTradeRequest request={};
   request.action=TRADE_ACTION_PENDING;         // definição de uma ordem pendente
   request.magic=magic_number;                  // ORDER_MAGIC
   request.symbol=_Symbol;                      // símbolo
   request.volume=0.1;                          // volume em 0.1 lotes
   request.sl=0;                                // Stop Loss (Parar Perda) não é espec
   request.tp=0;                                // Take Profit (Tomar Lucro) não é esp
//--- formar o tipo de ordem
   request.type=GetRandomType();                // tipo de ordem
//--- formar o preço para uma ordem pendente
   request.price=GetRandomPrice(request.type);  // preço de abertura
//--- enviar um pedido de negociação
   MqlTradeResult result={};
   OrderSend(request,result);
//--- escrever resposta do servido para log
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- código de retorno da resposta do servidor de negociação
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| tipo de retorno de uma ordem pendente de maneira aleatória       |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- valor incorreto
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//|  retorno do preço de forma aleatória                             |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- níveis de stop (parar) para símbolo
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- receber dados do último tick (ponto)
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- calcular o preço de acordo com o tipo
```

© 2000-2024, MetaQuotes Ltd.

```
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT ou ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // divergir do preço de compra
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else              // ORDER_TYPE_SELL_LIMIT ou ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // divergir do preço de venda
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Veja Também

Tipos de Operação de Negociação, Estrutura de Solicitação de Negociação, Estrutura de Solicitação de Verificação de Resultados, Estrutura de uma Solicitação de Resultado de Negociação

© 2000-2024, MetaQuotes Ltd.

OrderSendAsync

A função OrderSendAsync() é usada para a realização de operações de negócio assíncronos sem esperar a resposta do servidor de negócio a um pedido enviado. A função é projetado para negociação de alta freqüência, quando, nos termos do algoritmo de negociação é inaceitável perder tempo à espera de uma resposta do servidor.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // Estrutura solicitada
   MqlTradeResult&   result        // Resultado da estrutura
   );
```

Parâmetros

request

[in] Um ponteiro para uma estrutura do tipo MqlTradeRequestque descreve a ação comercial do cliente.

result

[in,out] Ponteiro para uma estrutura do tipo MqlTradeResultdescrevendo o resultado da operação comercial, no caso de uma realização bem sucedida (se verdadeiro é retornado).

Valor do Retorno

Retorna verdadeiro se o pedido é enviado para um servidor de negócio. No caso de o pedido não for enviado, ele retorna falsa. No caso de o pedido é enviado, na variável *resultado* o código de resposta contém o valor TRADE_RETCODE_PLACED (código 10008) – "order placed". Execução bem sucedida significa que apenas o fato de enviar, mas não dá qualquer garantia de que o pedido chegou ao servidor de comércio e foi aceito para processamento. Ao processar o pedido recebido, um servidor de de negócio envia uma resposta a um terminal do cliente notificando a alteração no estado atual de posições, ordens e promoções, o que leva à geração do evento de negociação.

O resultado da execução do pedido de negócio sobre o servidor enviado pela função OrderSend() pode ser controlada pelo handler (Manipulador) OnTradeTransaction. Deve ser observado que o handler (manipulador) OnTradeTransaction será chamado várias vezes durante a execução de um pedido de negócio.

Por exemplo, ao enviar uma ordem de compra de mercado, ela é tratada, uma ordem de compra apropriada é criada para a conta, a ordem é então executada e removida da lista de ordens em aberto, e então ela é adicionada ao histórico de ordens, uma apropriada operação (deal) é adicionada ao histórico e uma nova posição é criada. Função OnTradeTransaction será chamado para cada um desses eventos. Para obter alguns desses dados, os parâmetros da função deve ser analisada:

- **trans** - este parâmetro obtém a estrutura MqlTradeTransaction descrevendo uma transação de negociação aplicada a uma conta de negócio.
- **request** - este parâmetro recebe estruturaMqlTradeRequest descrevendo o pedido de negócio resultou em uma transação comercial.
- **result** - este parâmetro obtém a estrutura MqlTradeResult descrevendo o resultado da execução de uma solicitação de negociação.

Observação

© 2000-2024, MetaQuotes Ltd.

Em termos de objetivos e parâmetros, a função é semelhante ao OrderSend(), mas ao contrário, ela é assíncrona, isto é, não se sustenta o funcionamento do programa enquanto aguarda o resultado da execução da função. Você pode comparar a taxa de operações de comércio destas duas funções, utilizando a amostra Expert Advisor.

Exemplo:

```
#property description "Expert Advisor for sending trade requests "
                      " using OrderSendAsync() function.\r\n"
#property description "Handling trading events using"
                      " OnTrade() and OnTradeTransaction() handler functions is displa
#property description "Expert Advisor parameters allow setting Magic Number"
                      " (unique ID) "
#property description "and the mode of displaying messages in Experts log. All details
//--- parâmetros de entrada
input int  MagicNumber=1234567;      // Expert Advisor ID
input bool DescriptionModeFull=true; // Modo de saída detalhada
//--- chamada de variável para uso em HistorySelect()
datetime history_start;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificar se negociação automática (autotrading) é permitido
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("Autotrading in the terminal is disabled, Expert Advisor will be removed."
      ExpertRemove();
      return(-1);
     }
//--- inabilitada a negociação com uma conta real
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Expert Advisor cannot trade on a real account!");
      ExpertRemove();
      return(-2);
     }
//--- verificar se é possível negociar com esta conta (por exemplo, a negociação é imp
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Trading on this account is disabled");
      ExpertRemove();
      return(-3);
     }
//--- economizar o tempo do lançamento do Expert Advisor para receber histórico de neg
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- deletar todos os gráficos objetos
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| Função TradeTransaction                                          |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- título nomeado após negociação da função handler do evento
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- receber tipo de transação como valor de enumeração
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- se a transação é o resultado da solicitação de manipulação
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- visualizar nome da transação
      Print(EnumToString(type));
      //--- em seguida visualizar a descrição do string do pedido tratado
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- e mostrar descrição do resultado requisitado
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // visualizar a descrição completa da transação para as transações de outro t
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Função Trade                                                     |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- membros estáticos para o armazenamento de negociação do status da conta
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- histórico de negociação do pedido
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update)
//--- título nomeado após negociação da função handler do evento
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- exibir o nome do handler (manipulador) e o número de ordens no momento da manipu
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- exibir o número de ordens, as posições, promoções e alterações em parênteses
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- Inserir uma quebra de string para ver o log mais adequado
   Print("");
//--- salvar o status da conta
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| Função ChartEvent                                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- manipulação do evento CHARTEVENT_CLICK ("Clicando no gráfico")
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- volume mínimo de um negócio
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- Se o botão "Comprar" é pressionado, então comprar
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- desfazer pressionando o botão
```

```
          ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //--- Se o botão "Vender" é pressionado, então vender
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- desfazer pressionando o botão
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| Retorna o texto de descrição de uma transação                    |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- preparar uma string para o retorno a partir da função
   string desc=EnumToString(trans.type)+"\r\n";
//--- todos os dados possíveis é adicionado em modo detalhado
   if(detailed)
     {
      desc+="Ativo: "+trans.symbol+"\r\n";
      desc+="Bilhetagem (ticket) da operação: "+(string)trans.deal+"\r\n";
      desc+="Tipo de operação: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Bilhetagem (ticket) da ordem: "+(string)trans.order+"\r\n";
      desc+="Tipo de ordem: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Estado da ordem: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Ordem do tipo time: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Expiração da ordem: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Preço: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Gatilho do preço: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- retornar a string recebida
   return desc;
  }
//+-----------------------------------------------------------------+
//| Retorne o texto de descrição do resultado da manipulação de pedido |
//+-----------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- preparar uma string para o retorno a partir da função
```

© 2000-2024, MetaQuotes Ltd.

```
   string desc=EnumToString(request.action)+"\r\n";
//--- adicionar todos os dados disponíveis em modo detalhado
   if(detailed)
     {
      desc+="Ativo: "+request.symbol+"\r\n";
      desc+="Número mágico: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Bilhetagem (ticket) da ordem: "+(string)request.order+"\r\n";
      desc+="Tipo de ordem: "+EnumToString(request.type)+"\r\n";
      desc+="Preenchimento da ordem: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Ordem do tipo time: "+EnumToString(request.type_time)+"\r\n";
      desc+="Expiração da ordem: "+TimeToString(request.expiration)+"\r\n";
      desc+="Preço: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Pontos de desvio: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Comentário: "+request.comment+"\r\n";
     }
//--- retornar string recebida
   return desc;
  }
//+------------------------------------------------------------------+
//| Retorne o texto de descrição do resultado da manipulação de pedido |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- preparar a string para o retorno a partir da função
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- adicionar todos os dados disponíveis em modo detalhado
   if(detailed)
     {
      desc+="ID da Solicitação: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Bilhetagem (ticket) da ordem: "+(string)result.order+"\r\n";
      desc+="Bilhetagem (ticket) da operação: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Preço: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Compra (Ask): "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Venda (Bid): "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comentário: "+result.comment+"\r\n";
     }
//--- retornar string recebida
   return desc;
  }
//+------------------------------------------------------------------+
//| Criar dois botões de compra e venda                              |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- checar a presença do objeto chamado "Buy" (comprar)
   if(ObjectFind(0,"Buy")>=0)
     {
       //--- se o objeto encontrado não é um botão, deletar-lo
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // criar botão "Buy" (comprar)
//--- configurar botão "Buy" (comprar)
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- checar a presença do objeto chamado "Sell" (comprar)
   if(ObjectFind(0,"Sell")>=0)
     {
       //--- se o objeto encontrado não é um botão, deletar-lo
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // criar botão "Sell" (vender)
//--- configurar botão "Sell" (comprar)
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- realizar atualização forçada do gráfico para ver imediatamente os botões
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| Comprar usando a função OrderSendAsync() assíncrona              |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- preparar o pedido
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
```

```
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Comprar usando OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": erro ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
//| Vender usando a função OrderSendAsync() assíncrona               |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- preparar o pedido
   MqlTradeRequest req={};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_SELL;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation   =10;
   req.comment     ="Vende usando OrderSendAsync()";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": erro ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

Exemplo de exibição de mensagens em log "Experts":

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order filling: ORDER_FILLING_FOK
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Deviation points: 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment:
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
```

© 2000-2024, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000-2024, MetaQuotes Ltd.

PositionsTotal

Retorna o número de posições em aberto.

```
int  PositionsTotal();
```

Valor do Retorno

Valor do tipo int.

Observação

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Também Veja

PositionGetSymbol(), PositionSelect(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

PositionGetSymbol

Retorna o símbolo correspondente para a posição aberta e seleciona automaticamente a posição para posteriormente trabalhar com ele usando as funções PositionGetDouble, PositionGetInteger, PositionGetString.

```
string  PositionGetSymbol(
   int  index      // Número na lista de posições
   );
```

Parâmetros

index

[in] Número de posição na lista de posições abertas.

Valor do Retorno

Valor do tipo string. Se a posição não foi encontrada, uma string vazia será retornada. Para obter um código de erro, chamar a função GetLastError().

Observação

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Também Veja

PositionsTotal(), PositionSelect(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

PositionSelect

Escolhe uma posição aberta para posteriormente trabalhar com ele. Retorna true se a função for concluída com êxito. Retorna false em caso de falha. Para obter informações sobre o erro, chameGetLastError().

```
bool  PositionSelect(
   string  symbol      // Nome Simbolo
   );
```

Parâmetros

symbol

[in] Nome da segurança financeira.

Valor do Retorno

Valor do tipo bool.

Observação

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições. Neste caso, a PositionSelect escolhe a posição com o menor bilhete.

A função PositionSelect() copia os dados sobre a posição para o ambiente do programa, bem como outras chamadas de PositionGetDouble(), PositionGetInteger() e PositionGetString() retornar os dados copiados anteriormente. Isto significa que a posição em si já não podem existir (ou é o volume, orientação, etc mudou), mas ainda podem ser obtidos dados desta posição. Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar PositionSelect() logo antes referindo-se a eles.

Também Veja

PositionGetSymbol(), PositionsTotal(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

PositionSelectByTicket

Seleciona uma posição aberta para trabalhar no futuro com ela segundo o bilhete indicado. Retorna true se a função for bem-sucedida. Retorna false se a função falhar. Para obter informações sobre o erro, você deve chamar a função GetLastError().

```
bool  PositionSelectByTicket(
   ulong   ticket     // bilhete da posição
   );
```

Parâmetros

ticket

[in] Bilhete da posição.

valor de retorno

Valor do tipo bool.

Observação

A função PositionSelectByTicket() copia os dados da posição para o ambiente do sofware, e as chamadas subseqüentes da PositionGetDouble(), PositionGetInteger() e PositionGetString() retornar os dados de copiados anteriormente. Isto significa que a posição em si pode já não existir (ou ter alterado o volume, a direção, etc), e ainda pode receber os dados desta posição. Para garantir a recepção de novos dados sobre as posições, é recomendável chamar a função PositionSelectByTicket() imediatamente antes de se dirigir a elas.

Veja também

PositionGetSymbol(), PositionsTotal(), Propriedades das posições

© 2000-2024, MetaQuotes Ltd.

PositionGetDouble

A função retorna a propriedade solicitada de uma posição aberta, pré-selecionado usando PositionGetSymbol ou PositionSelect. A propriedade de negócio deve ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da execução da função. Se for bem sucedido, o valor da propriedade é colocada em uma variável de recebimento passada por referência no último parâmetro.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,      // Identificador de propriedade
   double&                        double_var        // Aqui nós aceitamos o valor da p
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade de operação. O valor pode ser um dos valores da enumeração ENUM_POSITION_PROPERTY_DOUBLE.

double_var

[out] Variável de tipo double, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo double. Se a função falhar, será retornado 0.

Observação

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar PositionSelect() logo antes referindo-se a eles.

Também Veja

PositionGetSymbol(), PositionSelect(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

PositionGetInteger

A função retorna a propriedade solicitada de uma posição aberta, pré-selecionado usando PositionGetSymbol ou PositionSelect. A propriedade de operação deve ser de tipo datetime, int. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da execução da função. Se for bem sucedido, o valor da propriedade é colocada em uma variável de recebimento passada por referência no último parâmetro.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // Identificador de propriedade
   long&                           long_var         // Aqui nós aceitamos o valor da p
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade de operação. O valor pode ser um dos valores de enumeração ENUM_POSITION_PROPERTY_INTEGER.

long_var

[out] Variável de tipo double, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo long. Se a função falhar, será retornado 0.

Observação

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar PositionSelect() logo antes referindo-se a eles.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função Trade                                                     |
//+------------------------------------------------------------------+
void OnTrade()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- verificar se a posição está presente e exibir a hora de sua mudança
   if(PositionSelect(_Symbol))
     {
receber ID posição para trabalhos futuros
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," posição #",position_ID);
//--- receber o tempo de posição formando em milissegundos desde 01.01.1970
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Posição #%d  POSITION_TIME_MSC = %i64 milissegundos => %s",positior
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- receber o tempo da última alteração da posição em segundos desde 01.01.1970
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Posição #%d  POSITION_TIME_UPDATE = %i64 segundos => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- receber o tempo da última alteração da posição em milissegundos desde 01.01.197
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Posição #%d  POSITION_TIME_UPDATE_MSC = %i64 milissegundos => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Também Veja

PositionGetSymbol(), PositionSelect(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

PositionGetTicket

A função retorna o bilhete da posição segundo o índice na lista de posições abertas e seleciona automaticamente esta posição para trabalhar com ela no futuro usando as funções PositionGetDouble, PositionGetInteger, PositionGetString.

```
ulong  PositionGetTicket(
   int  index      // número na lista de posições
   );
```

Parâmetros

index

[Em] Índice da posição na lista de posições abertas, começando com 0.

valor de retorno

Bilhete da posição. No caso de uma execução sem êxito retorna 0.

Observação

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Para garantir a recepção de novos dados sobre as posições, é recomendável chamar a função PositionSelect() imediatamente antes de se dirigir a elas.

Veja também

PositionGetSymbol(), PositionSelect(), Propriedades das posições

© 2000-2024, MetaQuotes Ltd.

PositionGetString

A função retorna a propriedade solicitada de uma posição aberta, pré-selecionado usando PositionGetSymbol ou PositionSelect. A propriedade de negócio deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da execução da função. Se for bem sucedido, o valor da propriedade é colocada em uma variável de recebimento passada por referência no último parâmetro.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // Identificador de propriedade
   string&                        string_var       // Aqui nós aceitamos o valor de p
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade de operação. O valor pode ser um dos valores da enumeração ENUM_POSITION_PROPERTY_STRING.

string_var

[out] Variável de tipo string, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo string. Se a função falhar, uma string vazia é retornada.

Observação

For the "netting" interpretation of positions (ACCOUNT_MARGIN_MODE_RETAIL_NETTING and ACCOUNT_MARGIN_MODE_EXCHANGE), only one position can exist for a symbol at any moment of time. This position is a result of one or more deals. Do not confuse positions with valid pending orders, which are also displayed on the Trading tab of the Toolbox window.

If individual positions are allowed (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), multiple positions can be open for one symbol.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar PositionSelect() logo antes referindo-se a eles.

Também Veja

PositionGetSymbol(), PositionSelect(), Propriedades de uma Posição

© 2000-2024, MetaQuotes Ltd.

OrdersTotal

Retorna o número de ordens.

```
int  OrdersTotal();
```

Valor do Retorno

Valor do tipo double.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente. Uma ordem é um pedido para conduzir a transação, enquanto que uma posição é um resultado de um ou mais negócios.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Também Veja

OrderSelect(), OrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

OrderGetTicket

Retorna o ticket de uma ordem correspondente, selecionando automaticamente a ordem para trabalhos posteriores usando funções.

```
ulong  OrderGetTicket(
   int  index      // Número na lista de posições
   );
```

Parâmetros

index

[in] Número de ordem na lista de ordens atuais.

Valor do Retorno

Valor do tipo ulong. Se a função falhar, será retornado 0.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente. Uma ordem é um pedido para conduzir a transação, enquanto que uma posição é um resultado de um ou mais negócios.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

OrderGetTicket() função copia os dados sobre uma ordem para o ambiente do programa, e posteriormente chamadas de OrderGetDouble(), OrderGetInteger(), OrderGetString() retornar os dados copiados anteriormente. Isto significa que a própria ordem pode já não existir (ou seu preço de abertura, Stop Loss/Take Profit, níveis de lucro ou expiração mudou), mas ainda podem ser obtidos dados dessa ordem. Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar PositionSelect() logo antes referindo-se a eles.

Exemplo:

```
void OnStart()
  {
//--- variáveis para retornar valores das propriedades de ordem
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- Numero de ordens atuais pendentes
   uint     total=OrdersTotal();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- passar por ordens em um loop
   for(uint i=0;i<total;i++)
     {
      //--- voltar ticket ordem por sua posição na lista
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- retorna propriedades de uma Ordem
         open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
         time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
         symbol        =OrderGetString(ORDER_SYMBOL);
         order_magic   =OrderGetInteger(ORDER_MAGIC);
         positionID    =OrderGetInteger(ORDER_POSITION_ID);
         initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
         type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
         //--- preparar e apresentar informações sobre a ordem
         printf("#ticket %d %s %G %s em %G foi criado em %s",
                ticket,                 // ticket de ordem
                type,                   // tipo
                initial_volume,         // volume colocado
                symbol,                 // simbolo
                open_price,             // preço de abertura especificada
                TimeToString(time_setup)// tempo de colocar a ordem
                );
        }
     }
//---
  }
```

Também Veja

OrdersTotal(), OrderSelect(), OrderGetInteger()

© 2000-2024, MetaQuotes Ltd.

OrderSelect

Seleciona uma ordem para trabalhar. Retorna verdadeiro se a função tenha sido concluída com êxito. Retorna false se a conclusão função falhou. Para mais informações sobre um erro, chame GetLastError().

```
bool  OrderSelect(
   ulong   ticket      // Ticket de ordem
   );
```

Parâmetros

ticket

[in] Ticket de ordem.

Valor do Retorno

Valor do tipo bool.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

A função OrderGetTicket() função copia os dados sobre uma ordem para o ambiente do programa, e posteriormente chamadas de OrderGetDouble(), OrderGetInteger(), OrderGetString() retornar os dados copiados anteriormente. Isto significa que a própria ordem pode já não existir (ou seu preço de abertura, Stop Loss/Take Profit, níveis de lucro ou expiração mudou), mas ainda podem ser obtidos dados dessa ordem. Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar OrderSelect() logo antes referindo-se a eles.

Também Veja

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

OrderGetDouble

Retorna a propriedade solicitada de uma ordem, pré-selecionado usando OrderGetTicket ou OrderSelect. A propriedade da ordem deve ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // Identificador de propriedade
   );
```

2. Retorna true ou false, dependendo do sucesso na execução da função. Se for bem sucedido, o valor da propriedade é colocado em uma variável alvo passado por referência até ao último parâmetro.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // Identificador de propriedade
   double&                     double_var         // Aqui nós aceitamos o valor da
   );
```

Parâmetros

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variável de tipo double, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo double. Se a função falhar, será retornado 0.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar OrderSelect() logo antes referindo-se a eles.

Também Veja

OrdersTotal(), OrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

OrderGetInteger

Retorna a propriedade solicitada de uma ordem, pré-selecionado usando OrderGetTicket ou OrderSelect. Propriedade de uma Ordem deve ser da tipo datetime, int. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // Identificador de propriedade
   long&                        long_var           // Aqui nós aceitamos o valor da p
   );
```

Parâmetros

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variável do tipo long que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo long. Se a função falhar, será retornado 0.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar OrderSelect() logo antes referindo-se a eles.

Também Veja

OrdersTotal(), OrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

OrderGetString

Retorna a propriedade solicitada de uma ordem, pré-selecionado usando OrderGetTicket ou OrderSelect. A propriedade da ordem deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,       // Identificador de propriedade
   string&                     string_var         // Aqui nós aceitamos o valor de pro
   );
```

Parâmetros

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variável de tipo string que aceita o valor da propriedade requerida...

Valor do Retorno

Valor do tipo string.

Observação

Não confunda atuais ordens pendentes com posições, que também são exibidos na aba "negócios" da "caixa de ferramentas" do terminal do cliente.

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições.

Para garantir o recebimento de novos dados sobre a posição, é recomendado chamar OrderSelect() logo antes referindo-se a eles.

Também Veja

OrdersTotal(), OrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistorySelect

Recupera o histórico de transações e ordens para o período do tempo de servidor especificado.

```
bool  HistorySelect(
   datetime  from_date,      // a partir da data
   datetime  to_date         // até à data
   );
```

Parâmetros

from_date

[in] Data de inicio do pedido.

to_date

[in] Data de término do pedido.

Valor do Retorno

Retorna verdadeiro true se bem sucedido, caso contrário false.

Observação

HistorySelect() cria uma lista de pedidos e uma lista de comércios em um programa de MQL5, para posterior referindo-se aos elementos da lista usando as funções correspondentes. O tamanho da lista de negócios pode ser devolvido através da função HistoryDealsTotal() o tamanho da lista de encomendas na história pode ser obtida usando HistoryOrdersTotal(). Seleção na lista de pedidos deve ser melhor realizado por HistoryOrderGetTicket(), para itens na lista de negócios HistoryDealGetTicket() se encaixarem melhor.

Depois de usar HistoryOrderSelect(), a lista de pedidos de história disponíveis para o programa MQL5 é reposto e completada novamente pela ordem encontrada, se a busca de uma ordem do ticket foi concluída com sucesso. O mesmo aplica-se à lista de ofertas disponíveis para o programa MQL5 - é reiniciado pelo HistoryDealSelect() e preenchido novamente no caso de uma recepção bem sucedida de um negócio pelo número do ticket.

Exemplo:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- história do negócio pedido
   HistorySelect(0,TimeCurrent());
//--- cria objetos
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
```

© 2000-2024, MetaQuotes Ltd.

```
   long     entry;
//--- para todos os negócios
   for(uint i=0;i<total;i++)
     {
      //--- tentar obter ticket negócios
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- obter as propriedades negócios
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- apenas para o símbolo atual
         if(price && time && symbol==Symbol())
           {
            //--- cria o preço do objeto
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- definir propriedades do objeto
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- aplicar no gráfico
   ChartRedraw();
  }
```

Também Veja

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2024, MetaQuotes Ltd.

HistorySelectByPosition

Recupera o histórico de transações e ordens para o período do tempo de servidor especificado.

```
bool  HistorySelectByPosition(
   long   position_id     // identificar posição - POSITION_IDENTIFIER
   );
```

Parâmetros

position_id

[in] Identificador de posição que está definido para cada ordem executada e cada negócio.

Valor do Retorno

Retorna verdadeiro true se bem sucedido, caso contrário false.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

HistorySelectByPosition() cria em um programa MQL5 uma lista de pedidos e uma lista de ofertas com um identificador de posição especificada para referência futura para os elementos da lista, utilizando as funções apropriadas. Para conhecer o tamanho da lista de promoções, utilizar a função HistoryDealsTotal(), o tamanho da lista de encomendas na história pode ser obtida usando HistoryOrdersTotal(). Para percorrer os elementos da lista de encomendas, utilize HistoryOrderGetTicket(), para os elementos da lista de ordens - HistoryDealGetTicket().

Depois de usar HistoryOrderSelect(), a lista do histórico de pedidos disponíveis para o programa MQL5 é reposto e completada novamente pela ordem encontrada, se a busca de uma ordem do ticket foi concluída com sucesso. O mesmo refere-se à lista de ofertas disponíveis para o programa MQL5 - é reposto pela função HistoryDealSelect() e é preenchido novamente se um acordo foi encontrado com sucesso pelo numero do ticket.

Também Veja

HistorySelect(), HistoryOrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrderSelect

Seleciona uma ordem a partir do histórico para posterior chamada através de funções apropriadas. Retorna true se a função tenha sido concluída com êxito. Retorna false se a função teve falha. Para maiores detalhes sobre erro chamar GetLastError().

```
bool  HistoryOrderSelect(
   ulong  ticket      // Ticket de ordem
   );
```

Parâmetros

ticket

[in] Ticket de ordem.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

HistoryOrderSelect() limpa em um programa de MQL5 a lista de ordens a partir de um histórico, disponível para chamadas, e copia ele para uma única ordem, se a execução de HistoryOrderSelect() foi concluída com êxito. Se você precisa passar por todas as negociações selecionadas pelo HistorySelect() , você deve usar melhor o HistoryOrderGetTicket().

Também Veja

HistorySelect(), HistoryOrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrdersTotal

Retorna o número de ordens no histórico. Antes de chamar HistoryOrdersTotal(), primeiro é necessário receber o histórico de negócios e ordens usando a função HistorySelect() ou a função HistorySelectByPosition().

```
int  HistoryOrdersTotal();
```

Valor do Retorno

Valor do tipo double.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

Também Veja

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetTicket

Retornar o ticket de uma ordem correspondente no histórico. Antes de chamar HistoryDealGetTicket(), primeiro é necessário receber o histórico de negócios e ordens utilizando a função HistorySelect() ou a função HistorySelectByPosition().

```
ulong  HistoryOrderGetTicket(
   int  index      // Número na lista de posições
   );
```

Parâmetros

index

[in] Número de ordens em uma lista de ordens.

Valor do Retorno

Valor do tipo ulong. Se a função falhar, será retornado 0.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

Exemplo:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- solicitar todo o histórico
   HistorySelect(from,to);
//--- variáveis para retornar valores das propriedades de ordem
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- Numero de ordens atuais pendentes
   uint     total=HistoryOrdersTotal();
//--- passar por ordens em um loop
   for(uint i=0;i<total;i++)
     {
      //--- voltar ticket ordem por sua posição na lista
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
```

© 2000-2024, MetaQuotes Ltd.

```
        //--- retorna propriedades de uma Ordem
        open_price=       HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
        time_setup=       (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
        time_done=        (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
        symbol=           HistoryOrderGetString(ticket,ORDER_SYMBOL);
        order_magic=      HistoryOrderGetInteger(ticket,ORDER_MAGIC);
        positionID =      HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
        initial_volume=   HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
        type=GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
        //--- preparar e apresentar informações sobre a ordem
        printf("#ticket %d %s %G %s at %G foi criado em %s => feito em %s, pos ID=%d"
               ticket,                  // ticket de ordem
               type,                    // tipo
               initial_volume,          // volume colocado
               symbol,                  // simbolo
               open_price,              // preço de abertura especificado
               TimeToString(time_setup),// tempo de colocar ordem
               TimeToString(time_done), // tempo de deletar ou executar a ordem
               positionID               // ID de uma posição, ao qual a quantidade de
               );
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Retorna o nome string do tipo de ordem                           |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="unknown operation";
   switch(type)
     {
      case (ORDER_TYPE_BUY):            return("compra");
      case (ORDER_TYPE_SELL):           return("vender");
      case (ORDER_TYPE_BUY_LIMIT):      return("buy limit");
      case (ORDER_TYPE_SELL_LIMIT):     return("sell limit");
      case (ORDER_TYPE_BUY_STOP):       return("buy stop");
      case (ORDER_TYPE_SELL_STOP):      return("sell stop");
      case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
      case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
     }
   return(str_type);
  }
```

Também Veja

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetDouble

Retorna a requisição de propriedade de ordem. A propriedade da ordem deve ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  HistoryOrderGetDouble(
   ulong                        ticket_number,      // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_DOUBLE   property_id         // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryOrderGetDouble(
   ulong                        ticket_number,      // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_DOUBLE   property_id,        // Identificar propriedade
   double&                      double_var          // Aqui nós aceitamos o valor da pro
   );
```

Parâmetros

ticket_number

[in] Ticket de ordem.

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_DOUBLE.

double_var

[out] Variável de tipo double, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo double.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

Também Veja

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetInteger

Retorna a propriedade solicitada de uma ordem. A propriedade de operação deve ser de tipo datetime, int. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,     // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,     // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // Identificador de propriedade
   long&                        long_var           // Aqui nós aceitamos o valor da pr
   );
```

Parâmetros

ticket_number

[in] Ticket de ordem.

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_INTEGER.

long_var

[out] Variável do tipo long que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo long.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função Trade                                                     |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- receber ticket da última ordem do histórico comercial da semana
   ulong last_order=GetLastOrderTicket();
```

© 2000-2024, MetaQuotes Ltd.

```
   if(HistoryOrderSelect(last_order))
     {
      //--- tempo de colocar um ordem em milissegundos desde 01.01.1970
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Ordem #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- execução/cancelamento de ordens em milissegundos desde 01.01.1970
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Ordem #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // notificação sobre falha
      PrintFormat("HistoryOrderSelect() falhou por #%d. Código de erro=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| Retorna o último ticket de ordem em histórico ou -1              |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
requisita o histórico para os últimos 7 dias
   if(!GetTradeHistory(7))
     {
      //--- notificar sobre chamada sem sucesso e retorno -1
      Print(__FUNCTION__," HistorySelect() retornou false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- trabalhar com ordens se houver alguma
   if(orders>0)
     {
      Print("Ordens = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- nenhuma ordem encontrada, retornar -1
   return -1;
  }
//+------------------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Solicitações de histórico para os últimos dias e retorna false em caso de falha |
//+------------------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- define um período da semana para solicitar histórico da operação
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- fazer um pedido e verificar o resultado
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Código de erro=",GetLastError());
      return false;
     }
//--- histórico recebido com sucesso
   return true;
  }
```

Também Veja

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryOrderGetString

Retorna a propriedade solicitada de uma ordem. A propriedade da ordem deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  HistoryOrderGetString(
   ulong                        ticket_number,      // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_STRING   property_id         // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryOrderGetString(
   ulong                        ticket_number,      // Ticket (bilhete)
   ENUM_ORDER_PROPERTY_STRING   property_id,        // Identificador de propriedade
   string&                      string_var          // Aqui nós aceitamos o valor de pro
   );
```

Parâmetros

ticket_number

[in] Ticket de ordem.

property_id

[in] Identificador da propriedade ordem. O valor pode ser um dos valores da enumeração ENUM_ORDER_PROPERTY_STRING.

string_var

[out] Variável do tipo string.

Valor do Retorno

Valor do tipo string.

Observação

Não confundir ordens de um histórico de negociação com as ordens pendentes que aparecem no separador "Comércio (Trade) da barra de "caixa de ferramentas" (Toolbox). A lista de ordens que foram cancelados ou levaram a uma transação, pode ser visto na aba "Histórico" da "caixa de ferramentas" do terminal do cliente.

Também Veja

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Propriedades de uma Ordem

© 2000-2024, MetaQuotes Ltd.

HistoryDealSelect

Seleciona um negócio no histórico para posterior chamada através de funções apropriadas. Retorna true se a função tenha sido concluída com êxito. Retorna false se a função teve falha. Para maiores detalhes sobre erro chamar GetLastError().

```
bool  HistoryDealSelect(
   ulong  ticket      // Ticket de negócio
   );
```

Parâmetros

ticket

[in] Ticket de operação.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

HistoryDealSelect() limpa a lista de negócios disponíveis para consulta em um programa-MQL5, e copia um único negócio, se a execução de HistoryDealSelect() foi concluída com êxito. Se você precisa passar por todos os negócios selecionados pela função HistorySelect(), é melhor você usar HistoryDealGetTicket().

Também Veja

HistorySelect(), HistoryDealGetTicket(), Propriedades de Negócio

© 2000-2024, MetaQuotes Ltd.

HistoryDealsTotal

Retorna o número de ordens no histórico. Retorna o número de negócio no histórico. Antes de chamar HistoryDealsTotal(), primeiro é necessário receber o histórico de negócios e ordens utilizando a função HistorySelect() ou a função HistorySelectByPosition().

```
int  HistoryDealsTotal();
```

Valor do Retorno

Valor do tipo double.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

Também Veja

HistorySelect(), HistoryDealGetTicket(), Propriedades de Negócio

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetTicket

A função seleciona um negócio para processamento posterior e retorna o ticket de negócio na história. Antes de chamar HistoryDealGetTicket (), primeiro é necessário receber o histórico de negócios e ordens utilizando a função HistorySelect() ou a função HistorySelectByPosition().

```
ulong  HistoryDealGetTicket(
   int  index      // ticket de negócio
   );
```

Parâmetros

index

[in] Número de um negócio na lista de negociações

Valor do Retorno

Valor do tipo ulong. Se a função falhar, será retornado 0.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

Exemplo:

```
void OnStart()
  {
   ulong deal_ticket;            // bilhetagem da operação (deal)
   ulong order_ticket;           // ticket da ordem que o negócio foi executado em
   datetime transaction_time;    // tempo de execução de um negócio
   long deal_type ;              // tipo de operação comercial
   long position_ID;             // ID posição
   string deal_description;      // descrição da operação
   double volume;                // volume da operação
   string symbol;                // ativo da negociação
//--- definir a data inicial e final para solicitar o histórico dos negócios
   datetime from_date=0;         // desde o princípio
   datetime to_date=TimeCurrent();//até o momento atual
//--- solicita o histórico das negociações no período especificado
   HistorySelect(from_date,to_date);
//--- número total na lista das negócios
   int deals=HistoryDealsTotal();
//--- agora processar cada trade (negócio)
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
```

© 2000-2024, MetaQuotes Ltd.

```
      position_ID=              HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=         GetDealDescription(deal_type,volume,symbol,order_tic
      //--- realizar uma boa formatação para o número de negócio
      string print_index=StringFormat("% 3d",i);
      //--- mostrar informações sobre o negócio
      Print(print_index+": deal #",deal_ticket," em ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//| Retorna a descrição string da operação                           |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("balance");
      case DEAL_TYPE_CREDIT:                   return ("credit");
      case DEAL_TYPE_CHARGE:                   return ("charge");
      case DEAL_TYPE_CORRECTION:               return ("correção");
      case DEAL_TYPE_BUY:                      descr="compra"; break;
      case DEAL_TYPE_SELL:                     descr="vender"; break;
      case DEAL_TYPE_BONUS:                    return ("bonus");
      case DEAL_TYPE_COMMISSION:               return ("comissão adicional");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("comissão diária");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("comissão mensal");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("comissão de agente diário");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("comissão de agente mensal");
      case DEAL_TYPE_INTEREST:                 return ("taxa de juros");
      case DEAL_TYPE_BUY_CANCELED:             descr="cancelado comprar negócio"; brea
      case DEAL_TYPE_SELL_CANCELED:            descr="cancelado vender negócio"; breal
     }
   descr=StringFormat("%s %G %s (ordem #%d, a posição ID %d)",
                      descr,  // descrição atual
                      volume, // volume de negócio
                      symbol, // ativo de negócio
                      ticket, // ticket da ordem que provocou o negócio
                      pos_ID  // ID de uma posição, na qual a negócio é incluído
                      );
   return(descr);
//---
  }
```

Também Veja

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriedades de uma Operação

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetDouble

Retorna a propriedade solicitada de uma operação. A propriedade de negócio deve ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_DOUBLE  property_id         // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,        // Identificador de propriedade
   double&                    double_var          // Aqui nós aceitamos o valor da prop
   );
```

Parâmetros

ticket_number

[in] Ticket de operação.

property_id

[in] Identificador de uma propriedade de operação. O valor pode ser um dos valores da enumeração ENUM_DEAL_PROPERTY_DOUBLE.

double_var

[out] Variável de tipo double, que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo double.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

Também Veja

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriedades de uma Operação

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetInteger

Retorna a propriedade solicitada de uma operação. A propriedade de operação deve ser de tipo datetime, int. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  HistoryDealGetInteger(
   ulong                       ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_INTEGER  property_id         // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryDealGetInteger(
   ulong                       ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_INTEGER  property_id,        // Identificador de propriedade
   long&                       long_var            // Aqui nós aceitamos o valor da pro
   );
```

Parâmetros

ticket_number

[in] Ticket (bilhete) de negócio.

property_id

[in] Identificador da propriedade de operação. O valor pode ser um dos valores da enumeração ENUM_DEAL_PROPERTY_INTEGER.

long_var

[out] Variável do tipo long que aceita o valor da propriedade requerida.

Valor do Retorno

Valor do tipo long.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

Exemplo:

```
//+------------------------------------------------------------------+
//| Função Trade                                                     |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- receber ticket da última operação do histórico comercial da semana
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- tempo de operações executadas em milissegundos desde 01.01.1970
       long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
       PrintFormat("Deal #%d DEAL_TIME_MSC=%i64 => %s",
                   last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
      }
    else
       PrintFormat("HistoryDealSelect() falhou por #%d. Código de erro=%d",
                   last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| Retorna o último bilhete de operação no histórico ou -1          |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
requisita o histórico para os últimos 7 dias
   if(!GetTradeHistory(7))
      {
       //--- notificar sobre chamada sem sucesso e retorno -1
       Print(__FUNCTION__," HistorySelect() retornou false");
       return -1;
      }
//---
   ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- trabalhar com ordens se houver alguma
   if(deals>0)
      {
       Print("Deals = ",deals);
       first_deal=HistoryDealGetTicket(0);
       PrintFormat("first_deal = %d",first_deal);
       if(deals>1)
         {
          last_deal=HistoryDealGetTicket((int)deals-1);
          PrintFormat("last_deal = %d",last_deal);
          return last_deal;
         }
       return first_deal;
      }
//--- nenhuma operação encontrada, retornar -1
   return -1;
  }
//+-----------------------------------------------------------------------------------+
//| Solicitações de histórico para os últimos dias e retorna false em caso de falha |
//+-----------------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- define um período da semana para solicitar histórico da operação
   datetime to=TimeCurrent();
```

```
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- fazer um pedido e verificar o resultado
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Código de erro=",GetLastError());
      return false;
     }
//--- histórico recebido com sucesso
   return true;
  }
```

Também Veja

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriedades de uma Operação

© 2000-2024, MetaQuotes Ltd.

HistoryDealGetString

Retorna a propriedade solicitada de uma operação. A propriedade de negócio deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  HistoryDealGetString(
   ulong                      ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_STRING  property_id         // Identificador de propriedade
   );
```

2. Retorna true ou false dependendo do sucesso da função. Se for bem sucedido, o valor da propriedade está situada dentro de uma variável de destino passado por referência pelo último parâmetro.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,      // Ticket (bilhete)
   ENUM_DEAL_PROPERTY_STRING  property_id,        // Identificador de propriedade
   string&                    string_var          // Aqui nós aceitamos o valor de prop
   );
```

Parâmetros

ticket_number

[in] Ticket de operação.

property_id

[in] Identificador da propriedade de operação. O valor pode ser um dos valores da enumeração ENUM_DEAL_PROPERTY_STRING.

string_var

[out] Variável de tipo string que aceita o valor da propriedade requerida...

Valor do Retorno

Valor do tipo string.

Observação

Não confunda ordens, operações e posições. Cada operação é o resultado da execução de uma ordem, cada posição é o resumo de um ou mais operações.

Também Veja

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Propriedades de uma Operação

© 2000-2024, MetaQuotes Ltd.

Gerenciamento de sinais

Grupo de funções para gerenciar sinais de negociação. Essas funções permitem:

- receber informações sobre os sinais de negociação disponíveis para cópia,
- ler ou configurar a cópia de sinais de negociação
- assinar um sinal, bem como a sua exclusão usando linguagem MQL5.

função	Ação
SignalBaseGetDouble	Retorna o valor da propriedade do tipo double para o sinal selecionado
SignalBaseGetInteger	Retorna o valor da propriedade do tipo integer para o sinal selecionado
SignalBaseGetString	Retorna o valor da propriedade do tipo string para o sinal selecionado
SignalBaseSelect	Seleciona o sinal na base de sinais de negociação disponíveis no terminal
SignalBaseTotal	Retorna o número total de sinais disponíveis no terminal
SignalInfoGetDouble	Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo double
SignalInfoGetInteger	Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo integer
SignalInfoGetString	Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo string
SignalInfoSetDouble	Define, nas configurações de cópia do sinal de negociação, o valor do tipo double
SignalInfoSetInteger	Define, nas configurações de cópia do sinal de negociação, o valor do tipo integer
SignalSubscribe	Realiza uma assinatura para copiar o sinal de negociação
SignalUnsubscribe	Cancela a assinatura para copiar o sinal de negociação

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetDouble

Retorna o valor da propriedade do tipo double para o sinal selecionado.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade do sinal. Pode ser um dos valores da enumeração ENUM_SIGNAL_BASE_DOUBLE.

valor de retorno

O valor do tipo double da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetInteger

Retorna o valor da propriedade do tipo integer para o sinal selecionado

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade do sinal. Pode ser um dos valores da enumeração ENUM_SIGNAL_BASE_INTEGER.

valor de retorno

O valor do tipo integer da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalBaseGetString

Retorna o valor da propriedade do tipo string para o sinal selecionado

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade do sinal. Pode ser um dos valores da enumeração ENUM_SIGNAL_BASE_STRING.

valor de retorno

O valor do tipo string da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalBaseSelect

Seleciona o sinal na base de sinais de negociação disponíveis no terminal.

```
bool  SignalBaseSelect(
   int     index     // índice de registro do sinal
   );
```

Parâmetros

index

[in] Índice de registro do sinal na base dos sinais de negociação.

valor de retorno

Retorna true se a função for bem-sucedida ou false no caso de erro. Para obter informações sobre o erro, você deve chamar a função GetLastError ().

Exemplo:

```
void OnStart()
  {
// --- pedimos o número total de sinais na base de dados
   int total=SignalBaseTotal();
// --- ciclo de acordo com todos os sinais
   for(int i=0;i<total;i++)
     {
      //--- selecionamos um sinal para o trabalho futuro
      if(SignalBaseSelect(i))
        {
         //--- obtenção das propriedades do sinal
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);          // id do sinal
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);        // resultado da
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // número de ass
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);         // nome do sinal
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);        // preço de assi
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);     // moeda do sina
         //--- exibimos todos os sinais rentáveis livres com zero de assinantes
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("Erro na seleção do sinal. Código de erro=%d",GetLastError());
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SignalBaseTotal

Retorna o número total de sinais disponíveis no terminal.

```
int  SignalBaseTotal();
```

valor de retorno

Número total de sinais disponíveis no terminal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetDouble

Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo double.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade para configurar a cópia do sinal de negociação.. Pode ser um dos valores da enumeração ENUM_SIGNAL_INFO_DOUBLE.

valor de retorno

O valor do tipo double da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetInteger

Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo integer.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade para configurar a cópia do sinal de negociação.. Pode ser um dos valores da enumeração ENUM_SIGNAL_INFO_INTEGER.

valor de retorno

O valor do tipo integer da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoGetString

Retorna, a partir das configurações de cópia do sinal de negociação, o valor do tipo string.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // identificador da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade para configurar a cópia do sinal de negociação.. O valor pode ser um dos valores da enumeração ENUM_SIGNAL_INFO_STRING.

valor de retorno

O valor do tipo string da propriedade indicada do sinal.

© 2000-2024, MetaQuotes Ltd.

SignalInfoSetDouble

Define, nas configurações de cópia do sinal de negociação, o valor do tipo double.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // identificador da propriedade
   double                       value            // valor da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade para configurar a cópia do sinal de negociação.. Pode ser um dos valores da enumeração ENUM_SIGNAL_INFO_DOUBLE.

value

[in] Valor da propriedade para configurar a cópia do sinal de negociação.

valor de retorno

Retorna true, se a alteração da propriedade for bem-sucedida, caso contrário, retorna false. Para mais informações sobre o erro, você deve chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalInfoSetInteger

Define, nas configurações de cópia do sinal de negociação, o valor do tipo integer.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // identificador da propriedade
   long                         value            // valor da propriedade
   );
```

Parâmetros

property_id

[in] Identificador da propriedade para configurar a cópia do sinal de negociação.. Pode ser um dos valores da enumeração ENUM_SIGNAL_INFO_INTEGER.

value

[in] Valor da propriedade para configurar a cópia do sinal de negociação.

valor de retorno

Retorna true, se a alteração da propriedade for bem-sucedida, caso contrário, retorna false. Para mais informações sobre o erro, você deve chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalSubscribe

Realiza uma assinatura para copiar o sinal de negociação.

```
bool  SignalSubscribe(
   long     signal_id     // id do sinal
   );
```

Parâmetros

signal_id

[in] Identificador do sinal.

valor de retorno

Retorna true, se a assinatura para copiar o sinal de negociação for bem-sucedida, caso contrário, retorna false. Para mais informações sobre o erro, você deve chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

SignalUnsubscribe

Cancela a assinatura para copiar o sinal de negociação

```
bool  SignalUnsubscribe();
```

valor de retorno

Retorna true, se a cancelação da assinatura do sinal de negociação for bem-sucedida, caso contrário, retorna false. false. Para mais informações sobre o GetLastError().

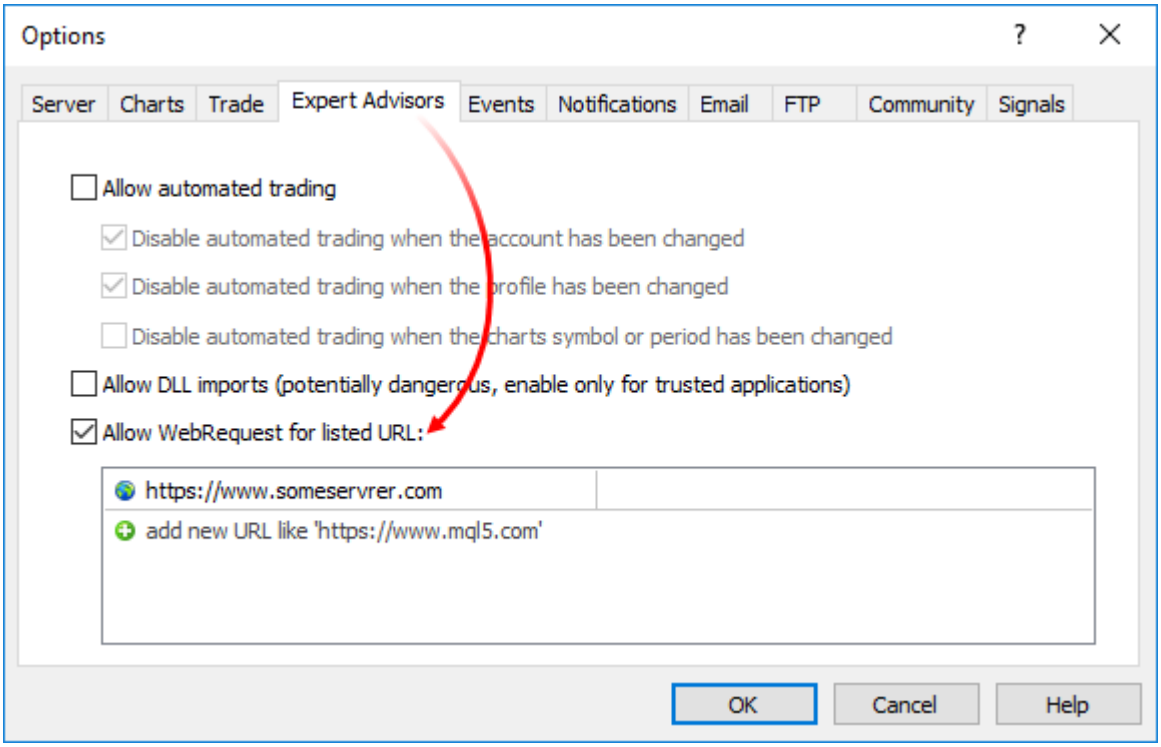
© 2000-2024, MetaQuotes Ltd.

Funções de rede

Os programas MQL5 podem trocar dados com servidores remotos, enviar notificações push, e-mails e dados FTP.

- O grupo de funções Socket* permite que você crie uma conexão TCP (incluindo uma conexão TLS segura) com um host remoto por meio de soquetes de sistema. O esquema de trabalho é simples: você cria o soquete, conecta-se ao servidor, e, assim, começaler e registrar dados.
- A função WebRequest é projetada para trabalhar com recursos da Web e facilita o envio de solicitações HTTP (incluindo GET e POST).
- SendFTP, SendMail e SendNotification são uma função mais simples para enviar arquivos, e-mail e notificações móveis.

Para segurança do usuário final, no lado do terminal do cliente é implementada uma lista de endereços IP permitidos, com a qual o programa MQL5 pode ser conectado usando as funções Socket* e WebRequest. Por exemplo, se ela precisar se conectar a https://www.someserver.com, esse endereço deverá ser explicitamente indicado pelo usuário do terminal na lista de permitidos. Via software não é possível adicionar o endereço.



Para notificar o usuário sobre a necessidade de configuração adicional, inclua uma mensagem explícita no programa MQL5. Por exemplo, via #property description, Alert ou Print.

Função	Ação
SocketCreate	Cria um soquete com os sinalizadores especificados e retorna seu identificador
SocketClose	Fecha o soquete
SocketConnect	Conecta-se ao servidor com controle de tempo limite
SocketIsConnected	Verifica se há um soquete conectado atualmente

© 2000-2024, MetaQuotes Ltd.

Função	Ação
SocketIsReadable	Obtém o número de bytes que podem ser lidos do soquete.
SocketIsWritable	Verifica se os dados podem ser registrados no soquete no momento atual.
SocketTimeouts	Define os tempos limite a fim de receber e enviar dados para o objeto de sistema de soquete
SocketRead	Lê dados do soquete
SocketSend	Registra dados no soquete
SocketTlsHandshake	Inicia uma conexão TLS (SSL) segura com o host especificado usando o protocolo TLS Handshake
SocketTlsCertificate	Recebe informações sobre o certificado usado para proteger a conexão de rede.
SocketTlsRead	Lê dados de uma conexão TLS segura
SocketTlsReadAvailable	Lê todos os dados disponíveis a partir de uma conexão TLS segura
SocketTlsSend	Envia dados por meio de uma conexão TLS segura
WebRequest	Envia uma solicitação HTTP para o servidor especificado
SendFTP	Envia um arquivo para o endereço especificado na janela de configurações na guia "FTP"
SendMail	Envia um e-mail para o endereço especificado na janela de configurações na guia "Mail"
SendNotification	Envia notificações por push para terminais móveis cuja ID MetaQuotes é indicada na guia "Notificações"

© 2000-2024, MetaQuotes Ltd.

SocketCreate

Cria um soquete com os sinalizadores especificados e retorna seu identificador.

```
int  SocketCreate(
   uint     flags         // sinalizadores
   );
```

Parâmetros

flags

[in] Combinação de sinalizadores definindo o modo de operação com o soquete. Atualmente, é suportado um sinalizador — SOCKET_DEFAULT.

Valor retornado

Após a criação bem-sucedida de um soquete, retorna seu identificador, caso contrário, INVALID_HANDLE.

Comentário:

Para liberar a memória do computador de um soquete não utilizado, chame para ele SocketClose.

De um programa MQL5 você pode criar um máximo de 128 soquetes. Quando exceder o limite em _LastError é registrado o erro 5271 (ERR_NETSOCKET_TOO_MANY_OPENED).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
```

```
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
     }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketClose

Fecha o soquete.

```
bool  SocketClose(
   const int  socket      // identificador do soquete
   );
```

Parâmetros

socket

[in] Identificador do soquete que deve ser fechado. O identificador é retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Observação

Se, para o soquete, anteriormente foi criada uma conexão SocketConnect, ela será interrompida.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
```

```
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
     }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketConnect

Conecta-se ao servidor com controle de tempo limite.

```
bool  SocketConnect(
   int           socket,              // soquete
   const string  server,              // endereço para conexão
   uint          port,                // porto para conexão
   uint          timeout_receive_ms   // tempo limite da conexão
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

server

[in] Nome de domínio do servidor ao qual você deseja se conectar ou seu endereço IP.

port

[in] Número da porta para conexão.

timeout_receive_ms

[in] Tempo limite de conexão em milissegundos. Se a conexão falhar durante esse período, serão interrompidas as tentativas.

Valor retornado

Retorna true na conexão bem-sucedida, caso contrário, false.

Observação

O endereço para conexão deve ser adicionado à lista de permitidos no lado do terminal do cliente (seção Serviços \ Configurações \ Expert Advisors).

Em caso de erro de conexão em _LastError é registrado o erro 5272 (ERR_NETSOCKET_CANNOT_CONNECT).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
```

```
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
```

© 2000-2024, MetaQuotes Ltd.

```
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
```

```
        Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
      }
    //--- fecha o soquete após ser usado
    SocketClose(socket);
   }
  else
    Print("Não foi possível criar o soquete, erro ",GetLastError());
 }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketIsConnected

Verifica se há um soquete conectado atualmente.

```
bool  SocketIsConnected(
   const int  socket      // identificador do soquete
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate(). Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true se o soquete estiver conectado, caso contrário, false.

Observação

Usando a função SocketIsConnected(), você pode verificar a conexão do soquete em um determinado momento.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

Veja também

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

© 2000-2024, MetaQuotes Ltd.

SocketIsReadable

Obtém o número de bytes que podem ser lidos a partir do soquete.

```
uint  SocketIsReadable(
   const int  socket      // identificador do soquete
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Número de bytes que podem ser lidos. Em caso de erro, retorna 0.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

Antes de chamar SocketRead, verifique se o soquete tem dados que ler. Caso contrário, na ausência de dados, a função SocketRead irá esperar em vão dados durante timeout_ms, atrasando a execução do programa.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
```

```
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
     }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketIsWritable

Verifica se os dados podem ser gravados no soquete no momento atual.

```
bool  SocketIsWritable(
   const int  socket      // identificador do soquete
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

Valor retornado

Retorna true se o registro for possível, caso contrário, false.

Observação

Usando esta função, você pode verificar se os dados podem ser gravados no soquete agora.

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

© 2000-2024, MetaQuotes Ltd.

SocketTimeouts

Define os tempos limite para receber e enviar dados para o objeto de sistema do soquete.

```
bool  SocketTimeouts(
   int           socket,               // soquete
   uint          timeout_send_ms,      // tempo limite para envio de dados
   uint          timeout_receive_ms    // tempo limite de aquisição de dados
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

timeout_send_ms

[in] Tempo limite de envio de dados em milissegundos.

timeout_receive_ms

[in] Tempo limite para receber dados em milissegundos.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Observação

Não confunda o tempo limite dos objetos do sistema e o tempo limite definido ao ler os dados via SocketRead. SocketTimeout define os tempos limite uma vez para o objeto do soquete no sistema operacional. Esses tempos limite serão aplicados a todas as funções de leitura e de envio de dados através desse soquete. Em SocketRead, o tempo limite é definido para uma operação de leitura de dados específica.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

© 2000-2024, MetaQuotes Ltd.

SocketRead

Lê dados do soquete.

```
int  SocketRead(
   int           socket,              // soquete
   uchar&        buffer[],            // buffer para ler dados do soquete
   uint          buffer_maxlen,       // número de bytes necessário para leitura
   uint          timeout_ms           // tempo limite de leitura
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Referência para array do tipo uchar em que serão lidos os dados. O tamanho da matriz dinâmica é aumentado pelo número de bytes lidos. O tamanho da matriz não pode exceder INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que devem ser lidos na matriz *buffer[]*. Os dados que não se encaixam na matriz permanecerão no soquete. Eles podem ser obtidos pela seguinte chamada SocketRead. O valor *buffer_maxlen* não pode exceder INT_MAX (2147483647).

timeout_ms

[in] Tempo limite de leitura de dados em milissegundos. Se os dados não puderem ser recuperados durante esse tempo, as tentativas serão concluídas e a função retornará -1.

Valor retornado

Se bem-sucedido, retorna o número de bytes lidos, em caso de erro, retorna -1.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

Se houver um erro ao ler os dados em _LastError é registrado o erro 5273 (ERR_NETSOCKET_IO_ERROR).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
```

```
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
```

```
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
     }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Veja também

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketSend

Registra dados no soquete.

```
int  SocketSend(
   int           socket,              // soquete
   const uchar&  buffer[],            // buffer para dados
   uint          buffer_len           // tamanho do buffer
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Referência para o array do tipo uchar com os dados que é preciso enviar para o soquete.

buffer_len

[in] Tamanho do array buffer.

Valor retornado

Se bem-sucedido, retorna o número de bytes registrados no socket. Em caso de erro, retorna -1.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

Se houver um erro ao registrar dados em _LastError é registrado o erro 5273 (ERR_NETSOCKET_IO_ERROR).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port    =80;
```

```
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("Cabeçalho HTTP de resposta recebido:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
```

```
        }
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
    else
       Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Veja também

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2024, MetaQuotes Ltd.

SocketTlsHandshake

Inicia uma conexão TLS (SSL) segura com o host especificado usando o protocolo TLS Handshake. Durante o Handshake, o cliente e o servidor concordam com os parâmetros de conexão: a versão do protocolo usado e o método de criptografia de dados.

```
bool  SocketTlsHandshake(
   int           socket,                // soquete
   const string  host                   // endereço do host
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

host

[in] Endereço do host com o qual é estabelecida uma conexão segura.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Comentário:

Antes de uma conexão segura, o programa deve estabelecer uma conexão TCP normal ao host usando SocketConnect.

Em caso de falha ao estabelecer uma conexão segura, em _LastError é registrado o erro 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED).

Não é necessária a chamada desta função, se a conexão é realizada para a porta 443. Essa é a porta TCP padrão usada para conexões TLS (SSL) seguras.

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

© 2000-2024, MetaQuotes Ltd.

SocketTlsCertificate

Obtém informações sobre o certificado usado para proteger uma conexão de rede.

```
int  SocketTlsCertificate(
   int           socket,              // soquete
   string&       subject,             // proprietário do certificado
   string&       issuer,              // emissor do certificado
   string&       serial,              // número de série do certificado
   string&       thumbprint,          // impressão digital do certificado
   datetime&     expiration           // data de expiração do certificado
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

subject

[in] Nome do proprietário do certificado. Corresponde ao campo Subject.

issuer

[in] Nome do emissor do certificado. Corresponde ao campo Issuer.

serial

[in] Número de série do certificado. Corresponde ao campo SerialNumber.

thumbprint

[in] Impressão digital do certificado. Corresponde ao hash SHA-1 do arquivo de certificado inteiro (todos os campos, incluindo a assinatura do editor).

expiration

[in] Data de expiração do certificado no formato datetime.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false.

Observação

A solicitação de dados do certificado só é possível após estabelecer uma conexão segura usando SocketTlsHandshake.

Ao falhar a obtenção do certificado, em _LastError é registrado o erro 5275 (ERR_NETSOCKET_NO_CERTIFICATE).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|                                                   SocketExample.mq5 |
//|                         Copyright 2018, MetaQuotes Software Corp. |
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
```

```
          if(rsp_len>0)
            {
             result+=CharArrayToString(rsp,0,rsp_len);
             //--- imprime apenas o cabeçalho da resposta
             int header_end=StringFind(result,"\r\n\r\n");
             if(header_end>0)
               {
                Print("Cabeçalho HTTP de resposta recebido:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
```

© 2000-2024, MetaQuotes Ltd.

```
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
            }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
         }
      else
         {
          Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
         }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
      }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
   }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

SocketTlsRead

Lê dados de uma conexão TLS segura

```
int  SocketTlsRead(
   int           socket,              // soquete
   uchar&        buffer[],            // buffer para ler dados do soquete
   uint          buffer_maxlen        // número de bytes necessário para leitura
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Referência para array do tipo uchar em que serão lidos os dados. O tamanho da matriz dinâmica é aumentado pelo número de bytes lidos. O tamanho da matriz não pode exceder INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que devem ser lidos na matriz *buffer[]*. Os dados que não se encaixam na matriz permanecerão no soquete. Eles podem ser recebidos pela seguinte chamada SocketTLSRead. O valor *buffer_maxlen* não pode exceder INT_MAX (2147483647).

Valor retornado

Se bem-sucedido, retorna o número de bytes lidos, em caso de erro, retorna -1.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

A função é executada até que receba a quantidade especificada de dados ou ocorra o tempo limite (SocketTimeouts).

Se houver um erro ao ler os dados em _LastError é registrado o erro 5273 (ERR_NETSOCKET_IO_ERROR).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 - "Função do sistema não permitida para chamada".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
```

```
#property version     "1.00"
#property description "Para o exemplo funcionar, adicione Address à lista de permitido
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Enviando comandos para o servidor                                |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- se for usada uma conexão TLS segura pela porta 443
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- se for usada uma conexão TCP normal
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Lendo a resposta do servidor                                     |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- lê dados do soquete enquanto eles existem, mas não mais tempo do que o timeout
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- diferentes comandos de leitura dependendo de se a conexão é segura ou n
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- analisa a resposta
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- imprime apenas o cabeçalho da resposta
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
```

© 2000-2024, MetaQuotes Ltd.

```
                {
                 Print("Cabeçalho HTTP de resposta recebido:");
                 Print(StringSubstr(result,0,header_end));
                 return(true);
                }
            }
        }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- verifica o identificador
   if(socket!=INVALID_HANDLE)
     {
      //--- se tudo estiver em ordem, conecte-se
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Conectado a ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- se a conexão estiver protegida por um certificado, exibe seus dados
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("Certificado TLS:");
            Print("   Proprietário:  ",subject);
            Print("   Emissor:  ",issuer);
            Print("   Número:     ",serial);
            Print("   Impressão digital: ",thumbprint);
            Print("   Expiração: ",expiration);
            ExtTLS=true;
           }
         //--- envia um pedido GET ao servidor
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("Solicitação GET enviada");
            //--- leia a resposta
            if(!HTTPRecv(socket,1000))
               Print("Falha ao obter resposta, erro ",GetLastError());
           }
         else
            Print("Falha ao enviar solicitação GET, erro ",GetLastError());
        }
```

```
      else
        {
         Print("Falhou conexão a ",Address,":",Port,", erro ",GetLastError());
        }
      //--- fecha o soquete após ser usado
      SocketClose(socket);
     }
   else
      Print("Não foi possível criar o soquete, erro ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Veja também

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketTlsReadAvailable

Lê todos os dados disponíveis a partir de uma conexão TLS segura.

```
int  SocketTlsReadAvailable(
   int           socket,              // soquete
   uchar&        buffer[],            // buffer para ler dados do soquete
   const uint    buffer_maxlen        // número de bytes necessário para leitura
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[out] Referência para array do tipo uchar em que serão lidos os dados. O tamanho da matriz dinâmica é aumentado pelo número de bytes lidos. O tamanho da matriz não pode exceder INT_MAX (2147483647).

buffer_maxlen

[in] Número de bytes que devem ser lidos no array buffer[]. Os dados que não se encaixam na matriz permanecerão no soquete. Eles estarão disponíveis chamando SocketTlsReadAvailable ou SocketTlsRead. O valor Buffer_maxlen não pode exceder INT_MAX (2147483647).

Valor retornado

Se bem-sucedido, retorna o número de bytes lidos, em caso de erro, retorna -1.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

Se houver um erro ao ler os dados em _LastError é registrado o erro 5273 (ERR_NETSOCKET_IO_ERROR).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

Veja também

SocketTimeouts, MathSwap

© 2000-2024, MetaQuotes Ltd.

SocketTlsSend

Envia dados por meio de uma conexão TLS segura.

```
int  SocketTlsSend(
   int           socket,              // soquete
   const uchar&  buffer[],            // buffer para dados
   uint          buffer_len           // tamanho do buffer
   );
```

Parâmetros

socket

[in] Identificador do soquete retornado pela função SocketCreate. Ao passar um identificador inválido para _LastError é registrado o erro 5270 (ERR_NETSOCKET_INVALIDHANDLE).

buffer

[in] Referência para o array do tipo uchar com os dados que é preciso enviar.

buffer_len

[in] Tamanho do array buffer.

Valor retornado

Se bem-sucedido, retorna o número de bytes registrados no socket. Em caso de erro, retorna -1.

Observação

Se ocorrer um erro durante a execução desta função no soquete do sistema, a conexão estabelecida Socketconnect será interrompida.

Se houver um erro ao registrar dados em _LastError é registrado o erro 5273 (ERR_NETSOCKET_IO_ERROR).

A função só pode ser chamada por EAs e scripts, pois eles trabalham em seu próprio fluxo de execução. Quando chamado do indicador GetLastError() retorna o erro 4014 – "Função do sistema não permitida para chamada".

Veja também

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2024, MetaQuotes Ltd.

WebRequest

A função envia uma solicitação HTTP para um servidor especificado. A função possui duas versões:

1. **Enviando solicitações simples** do tipo "key = value" usando o cabeçalho Content-Type: application/x-www-form-urlencoded.

```
int  WebRequest(
   const string      method,           // método HTTP
   const string      url,              // URL
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // tempo esgotado
   const char        &data[],          // A matriz do corpo da mensagem HTTP
   int               data_size,        // tamanho do array data[] em bytes
   char              &result[],        // Uma matriz contendo dados de resposta do se
   string            &result_headers   // cabeçalhos de resposta do servidor
   );
```

2. **Enviando uma solicitação de qualquer tipo** especificando o conjunto personalizado de cabeçalhos para uma interação mais flexível com vários serviços da Web.

```
int  WebRequest(
   const string      method,           // método HTTP
   const string      url,              // URL
   const string      headers,          // cabeçalho
   int               timeout,          // tempo esgotado
   const char        &data[],          // A matriz do corpo da mensagem HTTP
   char              &result[],        // Uma matriz contendo dados de resposta do se
   string            &result_headers   // cabeçalhos de resposta do servidor
   );
```

Parâmetros

método

[in] método HTTP.

url

[in] URL.

cabeçalho

[in] Cabeçalhos de solicitação do tipo "key: valor", separados por uma quebra de linha "\r\n".

cookie

[in] Valor do cookie.

referer

[in] Valor do cabeçalho Referer do pedido HTTP.

timeout

[in] Tempo limite em milissegundos.

© 2000-2024, MetaQuotes Ltd.

data[]

[in] Matriz de dados do corpo da mensagem HTTP.

data_size

[in] Tamanho do array data[].

result[]

[out] Um array contendo dados de resposta do servidor.

result_headers

[out] Cabeçalhos de resposta do servidor.

Valor retornado

Código de resposta do servidor HTTP ou -1 para um erro.

Nota

Para usar a função WebRequest(), adicione os endereços dos servidores necessários na lista de URLs permitidos na guia "Expert Advisors" da janela "Options". A porta do servidor é automaticamente selecionada com base no protocolo especificado - 80 para "http://" e 443 para "https://".

A função WebRequest() é síncrona, o que significa que interrompe a execução do programa e aguarda a resposta do servidor solicitado. Uma vez que os atrasos na recepção de uma resposta podem ser grandes, a função não está disponível para chamadas a partir dos indicadores, porque os indicadores são executados em uma linha comum compartilhada por todos os indicadores e gráficos em um símbolo. O atraso de desempenho do indicador em um dos gráficos de um símbolo pode interromper a atualização de todos os gráficos do mesmo símbolo.

A função pode ser chamada somente de Expert Advisors e scripts, como eles executam em seus próprios tópicos de execução. Se você tentar chamar a função de um indicador, GetLastError() retornará o erro 4014 - "Função não é permitida".

O WebRequest() não pode ser executada no Testador de Estratégia.

Um exemplo de usar a primeira versão da função WebRequest():

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- para trabalhar com o servidor é necessário adicionar a URL "https://finance.yaho
//--- na lista de URLs permitidas (menu Principal->Ferramentas->Opções, guia "Experts"
//--- redefinimos o código do último erro
   ResetLastError();
//--- download da página html do Yahoo Finance
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Erro no WebRequest. Código de erro =",GetLastError());
      //--- é possível que a URL não esteja na lista, exibimos uma mensagem sobre a ne
      MessageBox("É necessário adicionar um endereço '"+url+"' à lista de URL permitid
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  else
    {
     if(res==200)
       {
        //--- download bem-sucedido
        PrintFormat("O arquivo foi baixado com sucesso, tamanho %d bytes.",ArraySize
        //PrintFormat("Cabeçalhos do servidor: %s",headers);
        //--- salvamos os dados em um arquivo
        int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
        if(filehandle!=INVALID_HANDLE)
          {
           //--- armazenamos o conteúdo do array result[] no arquivo
           FileWriteArray(filehandle,result,0,ArraySize(result));
           //--- fechamos o arquivo
           FileClose(filehandle);
          }
        else
           Print("Erro em FileOpen. Código de erro =",GetLastError());
       }
     else
        PrintFormat("Erro de download '%s', código %d",url,res);
    }
}
```

© 2000-2024, MetaQuotes Ltd.

SendFTP

Envia um arquivo no endereço especificado na janela de configurações da guia "FTP".

```
bool  SendFTP(
   string  filename,          // nome do arquivo enviado
   string  ftp_path=NULL      // catálogo FTP
   );
```

Parâmetros

filename

[in] Nome do arquivo enviado.

ftp_path=NULL

[in] Catálogo FTP. Se um diretório não for especificado, o diretório descrito nas configurações é usado.

Valor do Retorno

Em caso de falha retorna false.

Observação

O arquivo enviado deve estar localizado na pasta *terminal_directory\MQL5\files* ou em suas subpastas. O envio não ocorre se o endereço FTP e/ou a senha de acesso não estiverem especificados nas configurações.

SendFTP() function does not work in the Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

SendMail

Envia um email no endereço especificado na janela de configurações da guia "Email".

```
bool  SendMail(
   string  subject,        // cabeçalho
   string  some_text       // texto de e-mail
   );
```

Parâmetros

subject

[in] Cabeçalho do email.

some_text

[in] Corpo do email.

Valor do Retorno

true – se um email é colocado na fila de envio, caso contrário - false.

Observação

O envio de emails pode ser proibido em configurações, o endereço de email pode ser omitido também. Para informações do erro chamar GetLastError().

SendMail() function does not work in the Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

SendNotification

Envia notificações push para terminais móveis, cujo MetaQuotes ID estão especificados na guia "Notificações".

```
bool  SendNotification(
   string  text          // Texto da notificação
   );
```

Parâmetros

text

[in] O texto da notificação. O comprimento da mensagem não deve exceder 255 caracteres.

Valor do Retorno

true se uma notificação foi enviada com sucesso do terminal; em caso de falha retorna false. Ao verificar após uma falha de envio de notificação, GetLastError () pode retornar um dos seguintes erros:

- 4515 – ERR_NOTIFICATION_SEND_FAILED,
- 4516 – ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 – ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 – ERR_NOTIFICATION_TOO_FREQUENT.

Observação

Rígidas restrições de uso estão definidas para a função SendNotification(): não mais que 2 chamadas por segundo e não mais que 10 chamadas por minuto. O monitoramento da freqüência de uso é dinâmico. A função pode ser desabilitada em caso de violação da restrição.

SendNotification() function does not work in the Strategy Tester.

© 2000-2024, MetaQuotes Ltd.

Variáveis Globais do Terminal de Cliente

Há um grupo que define as funções para trabalhar com variáveis globais.

Variáveis globais do terminal de cliente não deve ser confundida com as variáveis declaradas no escopo global do programa mql5.

Variáveis globais são guardadas no terminal de cliente por 4 semanas desde o último acesso, então elas serão excluídas automaticamente. Um acesso para a variável global não é somente a definição de um novo valor, mas a leitura do valor da varável global, também.

As variáveis globais do terminal de cliente são acessíveis simultaneamente de todos os programas mql5 executados no terminal do cliente.

Função	Ação
GlobalVariableCheck	Verifica a existência de uma variável global com o nome especificado
GlobalVariableTime	Retorna o tempo do último acesso para a variável global
GlobalVariableDel	Exclui uma variável global
GlobalVariableGet	Retorna o valor da variável global
GlobalVariableName	Retorna o nome de uma variável global, que é o número ordinal dentro da lista de variáveis globais
GlobalVariableSet	Define o novo valor para uma variável global
GlobalVariablesFlush	Salva forçadamente o conteúdo de todas as varáveis globais num disco
GlobalVariableTemp	Define o novo valor para uma variável global, que existe somente na sessão geral do terminal
GlobalVariableSetOnCondition	Define um novo valor de existência de variável global pela condição estabelecida
GlobalVariablesDeleteAll	Exclui as variáveis globais com especificação prefixada em seus nomes
GlobalVariablesTotal	Retorna o número total de Variáveis Globais

© 2000-2024, MetaQuotes Ltd.

GlobalVariableCheck

Verifica a existência de uma variável global com o nome especificado

```
bool  GlobalVariableCheck(
   string  name      // Nome da variável global
   );
```

Parâmetros

name

[in] Nome da variável global.

Valor do Retorno

Retorna true, se a variável global existe, caso contrário, retorna false.

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente.

Também Veja

GlobalVariableTime()

© 2000-2024, MetaQuotes Ltd.

GlobalVariableTime

Retorna o tempo em que a variável global foi acessado pela última vez.

```
datetime  GlobalVariableTime(
   string  name      // nome
   );
```

Parâmetros

name

[in] Nome da variável global.

Valor do Retorno

A função retorna hora do último acesso à variável global especificada. Chamando uma variável para obter um valor também é considerado como um acesso a ele. A fim de obter os detalhes do erro chamar a função GetLastError().

Observação

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização. Depois disso, eles são excluídos automaticamente.

Também Veja

GlobalVariableCheck()

© 2000-2024, MetaQuotes Ltd.

GlobalVariableDel

Exclui uma variável global a partir do terminal do cliente.

```
bool  GlobalVariableDel(
   string  name      // Nome da variável global
   );
```

Parâmetros

name

[in] Nome da variável global.

Valor do Retorno

Se bem sucedida, a função retorna true, caso contrário ela retorna false. Para obter uma informação sobre o erro é necessário chamar a função GetLastError().

Observação

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableGet

Retorna o valor de uma variável global existente do terminal do cliente. Existem 2 variantes da função.

1. Imediatamente retorna o valor da variável global.

```
double  GlobalVariableGet(
   string  name      // Nome da variável global
   );
```

2. Retorna true ou false dependendo do sucesso da execução da função. Se for bem sucedido, a variável global do terminal do cliente é colocado em uma variável passada por referência no segundo parâmetro.

```
bool  GlobalVariableGet(
   string  name,              // Nome da variável global
   double& double_var         // Esta variável irá conter o valor da variável global
   );
```

Parâmetros

name

[in] Nome da variável global.

double_var

[out] Alvo variável do tipo duplo, que aceita o valor armazenado em uma variável global do terminal do cliente.

Valor do Retorno

O valor da variável global existente ou 0(zero) em caso de um erro. Para mais detalhes sobre o erro, chamar GetLastError().

Observação

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableName

Retorna o nome de uma variável global, que é o número ordinal.

```
string  GlobalVariableName(
   int  index      // Número variável global na lista de variáveis globais
   );
```

Parâmetros

index

[in] Número de seqüência na lista de variáveis globais. Ele deve ser maior ou igual a 0(zero) e menor do que GlobalVariablesTotal().

Valor do Retorno

Nome da variável global, que é o número ordinal na lista de variáveis globais. Para mais detalhes sobre o erro, chamar GetLastError().

Observação

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableSet

Define um novo valor para uma variável global. Se a variável não existir, o sistema cria uma nova variável global.

```
datetime  GlobalVariableSet(
   string  name,       // Nome da variável global
   double  value       // Valor a ser definido
   );
```

Parâmetros

name

[in] Nome da variável global.

value

[in] O novo valor numérico.

Valor do Retorno

Se for bem sucedido, a função retorna a última modificação, caso contrário, 0 (zero). Para mais detalhes sobre o erro, chamar GetLastError().

Observação

Um nome de variável global não deve exceder 63 caracteres. As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesFlush

Salva forçadamente o conteúdo de todas as varáveis globais num disco.

```
void  GlobalVariablesFlush();
```

Valor do Retorno

Sem valor de retorno.

Observação

O terminal escreve todas as variáveis globais quando o trabalho é em excesso, mas os dados podem ser perdidos em uma falha súbita de operação do computador. Esta função permite controle independente do processo de salvar variáveis globais em caso de contingência.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableTemp

A função tenta criar uma variável global temporária. Se a variável não existir, o sistema cria uma nova variável global temporária.

```
bool  GlobalVariableTemp(
   string  name      // Nome da variável global
   );
```

Parâmetros

name

[in] O nome de uma variável global temporária.

Valor do Retorno

Se for bem sucedido, a função retorna true, caso contrário - false. Para obter detalhes sobre o erro, você deve chamar a função GetLastError().

Observação

Variáveis globais temporários existem apenas enquanto o terminal do cliente está sendo executado, após o desligamento do terminal eles são excluídos automaticamente. Note-se que durante a execução de GlobalVariablesFlush() variáveis globais temporárias não são gravados em um disco.

Depois de uma variável global temporário tiver sido criado, pode ser acedido e modificado a mesma variável global do terminal do cliente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariableSetOnCondition

Define o novo valor da variável global existente, se o valor da corrente é igual ao terceiro parâmetro check_value. Se não houver nenhuma variável global, a função irá gerar um erro ERR_GLOBALVARIABLE_NOT_FOUND (4501) e voltar false.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // Nome da variável global
   double  value,           // Novo valor para a variável, se a condição for verdadei
   double  check_value      // Verifique condição de valor
   );
```

Parâmetros

name

[in] O nome de uma variável global.

value

[in] Novo valor.

check_value

[in] O valor para verificar o valor atual da variável global.

Valor do Retorno

Se bem sucedida, a função retorna true, caso contrário ela retorna false. Para obter detalhes sobre o erro chamar GetLastError(). Se o valor atual da variável global é diferente de check_value, a função retorna false.

Observação

Função fornece acesso atômico a variável global, para que ele possa ser utilizado para a prestação de uma exclusão mútua na interação de vários consultores especializados trabalhando simultaneamente dentro de um terminal de cliente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesDeleteAll

Apaga as variáveis globais do terminal do cliente.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL,     // Todas as variáveis globais com nomes que começa
   datetime   limit_data=0          // Todas as variáveis globais que foram alteradas
   );
```

Parâmetros

prefix_name=NULL

[in] Prefixo do nome de variáveis globais para remover. Se você especificar um prefixo NULL ou string vazia, então todas as variáveis que atendam o critério dados serão apagados.

limit_data=0

[in] Data para selecionar as variáveis globais no momento de sua última modificação. A função remove variáveis globais, que foram alteradas antes desta data. Se o parâmetro for zero, então todas as variáveis que atendam o primeiro critério (prefixo) são excluídos.

Valor do Retorno

O número de variáveis apagados.

Observação

Se ambas as opções são iguais a zero (prefix_name = NULL e limit_data = 0), então função exclui todas as variáveis globais da terminação. Se ambos os parâmetros são especificados, então ele elimina variáveis globais correspondentes a ambos os parâmetros.

As variáveis globais existem no terminal do cliente durante 4 semanas desde suas última utilização, então eles são excluídos automaticamente.

© 2000-2024, MetaQuotes Ltd.

GlobalVariablesTotal

Retorna o número total de Variáveis Globais do Terminal de Cliente.

```
int  GlobalVariablesTotal();
```

Valor do Retorno

Número de variáveis globais.

Observação

As variáveis globais existem no terminal do cliente durante 4 semanas desde a última utilização, então eles são excluídos automaticamente. Chamada de uma variável global está a definir não só um novo valor, mas também a leitura do valor da variável global.

© 2000-2024, MetaQuotes Ltd.

Funções de Arquivo

Este é um grupo de funções para trabalhar com arquivos.

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Existem dois diretórios(com subdiretórios) onde os arquivos de trabalho estão localizados:

- terminal_data_folder\MQL5\FILES\ (no menu do terminal selecione para ver "Arquivo" - "Abrir o diretório de dados");
- na pasta comum para todos os terminais instalados no computador - geralmente no diretório C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files (C:\Documentos e Configurações\Todos Usuários\Dados de Aplicativos\MetaQuotes\Comum\Files).

Existe um método de programa para obter nome destes catálogos usando a função TerminalInfoString(), usando a enumeração ENUM_TERMINAL_INFO_STRING:

```
//--- Pasta que armazena os dados do terminal
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Pasta comum para todos os terminais de clientes
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Trabalhar com arquivos de outros diretórios é proibido.

Funções de arquivo permitem trabalhar com os assim chamados "pipes nomeados". Para isto, chamar simplesmente a função FileOpen() com o parâmetro apropriado.

Função	Ação
FileSelectDialog	Cria uma caixa de diálogo para abrir/criar um arquivo ou pasta
FileFindFirst	Começa a busca de arquivos em um diretório de acordo com o filtro especificado
FileFindNext	Continua a busca iniciada pela função FileFindFirst()
FileFindClose	Fecha o manipulador de busca
FileOpen	Abre um arquivo com o nome e bandeira especificada
FileDelete	Deleta um arquivo especificado
FileFlush	Escreve para um disco todos os dados remanescentes dentro da entrada/saída do arquivo buffer
FileGetInteger	Obtém uma propriedade inteira de um arquivo
FileIsEnding	Define o final do arquivo em processo de leitura
FileIsLineEnding	Define o final da linha do arquivo de texto em processo de leitura
FileClose	Fecha um arquivo aberto previamente
FileIsExist	Verifica a e existência de um arquivo

© 2000-2024, MetaQuotes Ltd.

Função	Ação
FileCopy	Copia o arquivo original a partir de um local ou de uma pasta compartilhada em outro arquivo
FileMove	Move ou renomeia um arquivo
FileReadArray	Lê arrays de qualquer tipo, exceto para a seqüência a partir do arquivo do tipo BIN
FileReadBool	Lê a partir de arquivo do tipo CSV uma seqüência a partir da posição atual até um delimitador (ou até o fim de uma linha de texto) e converte a seqüência de leitura para um valor do tipo bool
FileReadDatetime	Lê a partir do arquivo do tipo CSV uma seqüência de um dos formatos: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" or "HH:MM:SS" - e converte-os dentro de valor de datetime
FileReadDouble	Lê um valor double a partir da posição atual do ponteiro de arquivo
FileReadFloat	Lê um valor float a partir da posição atual do ponteiro de arquivo
FileReadInteger	Lê um valor int, short ou char a partir da posição atual do ponteiro de arquivo
FileReadLong	Lê um valor do tipo long a partir da posição atual do ponteiro de arquivo
FileReadNumber	Lê a partir de arquivo do tipo CSV uma seqüência a partir da posição atual até um delimitador (ou até o fim de uma linha de texto) e converte a seqüência de leitura em um valor double
FileReadString	Lê uma seqüência a partir da posição atual do ponteiro a partir de um arquivo
FileReadStruct	Lê o conteúdo de um arquivo binário em uma estrutura passada como um parâmetro, a partir da posição atual do ponteiro do arquivo
FileSeek	Move a posição do ponteiro de arquivo por um determinado número de bytes em relação à posição especificada
FileSize	Retorna o tamanho da abertura de um arquivo correspondente
FileTell	Retorna a posição atual do ponteiro de um arquivo da abertura de arquivo correspondente
FileWrite	Escreve dados para um arquivo de CSV ou tipo TXT
FileWriteArray	Escreve arrays de qualquer tipo exceto para seqüência em um arquivo do tipo BIN
FileWriteDouble	Escreve valor do tipo double a partir da posição atual de um ponteiro de arquivo em um arquivo binário
FileWriteFloat	Escreve o valor do tipo float a partir da posição atual de um ponteiro de arquivo em um arquivo binário

© 2000-2024, MetaQuotes Ltd.

Função	Ação
FileWriteInteger	Escreve o valor do tipo int a partir da posição atual de um ponteiro de arquivo em um arquivo binário
FileWriteLong	Escreve o valor do tipo long a partir da posição atual de um ponteiro de arquivo em um arquivo binário
FileWriteString	Escreve o valor de um parâmetro da string em um arquivo BIN ou TXT iniciando a partir da posição atual do ponteiro de arquivo
FileWriteStruct	Escreve o conteúdo de uma estrutura passada como um parâmetro em um arquivo binário, começando a partir da posição atual do ponteiro de arquivo
FileLoad	Ele lê todo o conteúdo do arquivo binário especificado numa variedade transferida de tipos numéricos ou estruturas simples
FileSave	Armazena no arquivo binário todos os elementos da matriz transferida como parâmetro
FolderCreate	Cria uma pasta dentro do diretório de arquivo
FolderDelete	Remove um diretório selecionado. Se a pasta não estiver vazia, então ele não pode ser removido
FolderClean	Deleta todos os arquivos dentro da pasta especificada

Se o arquivo é aberto para escrever usando FileOpen(),todas as sub-pastas especificadas no caminho serão criadas se elas não existirem.

© 2000-2024, MetaQuotes Ltd.

FileSelectDialog

Cria uma caixa de diálogo para abrir/criar um arquivo ou pasta.

```
int  FileSelectDialog(
   string   caption,              // barra de título
   string   initial_dir,          // pasta inicial
   string   filter,               // filtro de extensão
   uint     flags,                // combinação de sinalizadores
   string&  filenames[],          // matriz contendo os nomes dos arquivos
   string   default_filename      // nome do arquivo por padrão
   );
```

Parâmetros

caption

[in] Título da caixa de diálogo.

initial_dir

[in] Nome da pasta inicial relativa à pasta MQL5\Files, cujo conteúdo será exibido na caixa de diálogo. Se o valor for igual a NULL, na caixa de diálogo será mostrada a pasta MQL5\Files.

filter

[in] Filtro de extensões de arquivos que serão mostradas na caixa de diálogo para serem selecionadas. Arquivos de outros formatos ficarão ocultos.

flags

[in] Combinação de sinalizadores que define o modo da caixa de diálogo. Os sinalizadores são definidos da seguinte maneira:
FSD_WRITE_FILE - diálogo de abertura de arquivo
FSD_SELECT_FOLDER - permite selecionar apenas pastas;
FSD_ALLOW_MULTISELECT - permite selecionar alguns arquivos;
FSD_FILE_MUST_EXIST - os arquivos selecionados devem existir;
FSD_COMMON_FOLDER - o arquivo está localizado na pasta compartilhada de terminais do cliente \Terminal\Common\Files.

filenames[]

[out] Matriz de linhas na qual serão colocados os nomes dos arquivos/pastas selecionados.

default_filename

[in] Nome do arquivo/pasta por padrão. Se especificado, esse nome será automaticamente substituído no diálogo de abertura e retornado na matriz *filenames[]* durante o teste.

Valor retornado

Se for bem-sucedida, a função retornará o número de arquivos selecionados cujos nomes podem ser obtidos em *filenames[]*. Se o usuário não selecionar o arquivo e fechar a caixa de diálogo, a função retornará 0. Em caso de execução mal sucedida retornará um valor menor que 0, código de erro poderá ser obtido com a ajuda de GetLastError().

Observação

© 2000-2024, MetaQuotes Ltd.

Por motivos de segurança, em MQL5 o trabalho com arquivos é estritamente controlado. Os arquivos com os quais são executadas as operações de arquivo através da linguagem MQL5 não podem estar localizados fora do arquivo "área restrita", isto é, fora da pasta MQL5\Files.

O nome da pasta inicial*initial_dir* é procurado na pasta do terminal do cliente na subpasta MQL5\Files (ou diretório_agente_teste\MQL5\Files em caso de teste). Se entre os sinalizadores for especificado FSD_COMMON_FOLDER, a pasta inicial será procurada na pasta compartilhada de terminais do cliente \Terminal\Common\Files.

O parâmetro *filter* indica arquivos válidos e deve ser definido com o formato "<descrição 1>|<extensão 1>|<descrição 2>|<extensão 2>...". Por exemplo, "Text files (*.txt)|*.txt|All files (*.*)|*.*", além disso, a primeira extensão "Text files (*.txt)|*.txt" será selecionada como tipo de arquivo por padrão.
Se *filter*=NULL, a máscara de seleção de arquivos na caixa de diálogo será "All Files (*.*)|*.*|"

Se definido o parâmetro *default_filename*, então durante o teste não visual a chamada de FileSelectDialog() retornará 1, e o valor de *default_filename* será copiado na matriz *filenames[]*.

A função é proibida em indicadores personalizados, uma vez que a chamada para FileSelectDialog() interrompe o trabalho do thread de execução durante todo o tempo de espera pela resposta do usuário. Como todos os indicadores de cada símbolo são executados num único thread, o funcionamento de todos os gráficos em todos os períodos gráficos do símbolo em questão será interrompido.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- obtemos os nomes dos arquivos de texto para carregamento a partir da pasta compa
   string filenames[];
   if(FileSelectDialog("Selecionar arquivos para carregamento", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- exibimos o nome de cada arquivo selecionado
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Files not selected");
     }
//---
  }
```

Veja também

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Sinalizadores de abertura de arquivos

© 2000-2024, MetaQuotes Ltd.

FileFindFirst

A função começa a busca de arquivos e subdiretórios em um diretório de acordo com o filtro especificado.

```
long  FileFindFirst(
   const string   file_filter,          // String - filtro de pesquisa
   string&        returned_filename,    // Nome do arquivo ou subdiretório encontrado
   int            common_flag=0         // Define a pesquisa
   );
```

Parâmetros

file_filter

[in] Filtro de pesquisa. Um subdiretório (ou seqüência de subdiretórios aninhados) relativo ao diretório \Files, em que os arquivos devem ser procurados, podem ser especificados no filtro.

returned_filename

[out] O parâmetro retornado, onde, em caso de sucesso, o nome do primeiro arquivo encontrado ou subdiretório é colocado. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

common_flag

[in] Bandeira determinando a localização do arquivo. Se common_flag = FILE_COMMON, então o arquivo está localizado em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files. Caso contrário, o arquivo está localizado em uma pasta local.

Valor do Retorno

Retorna o manipulador do objeto pesquisado, que deve ser utilizado para posterior triagem dos arquivos e subdiretórios pela função FileFindNext(), ou INVALID_HANDLE quando não há nenhum arquivo e subdiretório correspondente ao filtro (no caso particular - quando o diretório está vazio. Após a pesquisa, o manipulador deve ser fechado usando a função FileFindClose().

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
//--- display the window of input parameters when launching the script
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- search for the last backslash
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- get the search handle in the root of the local folder
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- check if the FileFindFirst() is executed successfully
   if(search_handle!=INVALID_HANDLE)
     {
      //--- in a cycle, check if the passed strings are the names of files or directo
      do
        {
         ResetLastError();
         //--- if it's a file, the function returns true, and if it's a directory, it
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "D
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- close the search handle
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

Também Veja

FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

FileFindNext

A função continua a busca iniciada por FileFindFirst().

```
bool  FileFindNext(
   long      search_handle,         // Manipular pesquisa
   string&   returned_filename      // Nome do arquivo ou subdiretório encontrado
   );
```

Parâmetros

search_handle

[in] Manipulador de pesquisa, recuperado por FileFindFirst().

returned_filename

[out] O nome do próximo arquivo ou subdiretório encontrado. Only the file name is returned (including the extension), the directories and subdirectories are not included no matter if they are specified or not in the search filter.

Valor do Retorno

Se bem sucedido retorna verdadeiro, caso contrário false.

Exemplo:

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- receber o manipulador de pesquisa na raiz da pasta local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- verificar se FileFindFirst() é executada com sucesso
   if(search_handle!=INVALID_HANDLE)
     {
      //--- verificar se as strings passadas são nomes de arquivo ou diretório no loop
      do
        {
         ResetLastError();
         //--- se este é um arquivo, a função retornará verdadeiro, se é um diretório,
         FileIsExist(file_name);
         PrintFormat("%d : %s nome = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Fechar manipulador de pesquisa
      FileFindClose(search_handle);
    }
  else
      Print("Arquivo não encontrado!");
}
```

Também Veja

FileFindFirst, FileFindClose

© 2000-2024, MetaQuotes Ltd.

FileFindClose

A função fecha o identificador pesquisa.

```
void  FileFindClose(
   long  search_handle      //  Manipular pesquisa
   );
```

Parâmetros

search_handle

[in] Manipulador de pesquisa, recuperado por FileFindFirst().

Valor do Retorno

Sem valor retornado.

Observação

Função deve ser chamado para liberar recursos do sistema.

Exemplo:

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- filtro
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- receber o manipulador de pesquisa na raiz da pasta local
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- verificar se FileFindFirst() é executada com sucesso
   if(search_handle!=INVALID_HANDLE)
     {
      //--- verificar se as strings passadas são nomes de arquivo ou diretório no loo
      do
        {
         ResetLastError();
         //--- se este é um arquivo, a função retornará verdadeiro, se é um diretório,
         FileIsExist(file_name);
         PrintFormat("%d : %s nome = %s",i,GetLastError()==5018 ? "Diretório" : "Arqu
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- Fechar manipulador de pesquisa
      FileFindClose(search_handle);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    else
       Print("Arquivo não encontrado!");
   }
```

Também Veja

FileFindFirst, FileFindNext

© 2000-2024, MetaQuotes Ltd.

FileIsExist

Verifica a existência de um arquivo.

```
bool  FileIsExist(
   const string  file_name,        // Nome do arquivo
   int           common_flag=0     // Área de pesquisa
   );
```

Parâmetros

file_name

[in] O nome do arquivo a ser verificado

common_flag=0

[in] Bandeira determinando a localização do arquivo. Se common_flag = FILE_COMMON, então o arquivo está localizado em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files. Caso contrário, o arquivo está localizado em uma pasta local.

Valor do Retorno

Retorna true, se o arquivo especificado existe.

Observação

Arquivo verificado pode vir a ser um subdiretório. Neste caso, FileIsExist() função retornará falso, enquanto o erro 5018 será registrada na variável _LastError - "Este é um diretório, não é um arquivo" (ver exemplo pela função FileFindFirst).

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Se common_flag = FILE_COMMON, então a função procura o arquivo em uma pasta compartilhada por todos os terminais do cliente \Terminal\Common\Files, caso contrário, a função procura por um arquivo em uma pasta local (MQL5\Files ou MQL5\Tester\Files no caso de testes).

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- data para arquivos antigos
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // variável para armazenar os nomes dos arquivos
   string   filter="*.txt"; // filtrar parar pesquisar os arquivos
   datetime create_date;     // data de criação do arquivo
   string   files[];         // lista de nome dos arquivos
   int      def_size=25;     // tamanho do array por padrão
```

© 2000-2024, MetaQuotes Ltd.

```
   int      size=0;         // número de arquivos
//--- alocar memória para o array
   ArrayResize(files,def_size);
//--- receber o manipulador de pesquisa na raiz da pasta local
   long search_handle=FileFindFirst(filter,file_name);
//--- verificar se FileFindFirst() é executada com sucesso
   if(search_handle!=INVALID_HANDLE)
     {
      //--- pesquisando arquivos no loop
      do
        {
         files[size]=file_name;
         //--- aumentar o tamanho do array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- redefinir o valor de erro
         ResetLastError();
         //--- receber a data de criação do arquivo
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- verificar se o arquivo é antigo
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s arquivo deletado!",file_name);
            //--- deletado o arquivo antigo
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fechar o manipulador de pesquisa
      FileFindClose(search_handle);
     }
   else
     {
      Print("Arquivo não encontrado!");
      return;
     }
//--- verificar quais os arquivos permanecem
   PrintFormat("Resultados:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s arquivo existe!",files[i]);
      else
         PrintFormat("%s arquivo deletado!",files[i]);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

Também Veja

FileFindFirst

© 2000-2024, MetaQuotes Ltd.

FileOpen

A função abre o arquivo com o nome e a bandeira especificada.

```
int  FileOpen(
   string  file_name,          // Nome do arquivo
   int     open_flags,         // Combinação de bandeiras
   short   delimiter='\t',     // Delimitador
   uint    codepage=CP_ACP     // Página de código
   );
```

Parâmetros

file_name

[in] O nome do arquivo pode conter sub-pastas. Se o arquivo é aberto para escrita, estas sub-pastas serão criadas se elas não existirem.

open_flags

[in] combinação de bandeiras que determinam o modo de operação para o arquivo. As bandeiras são definidas das seguintes formas:
FILE_READ arquivo é aberto para leitura
FILE_WRITE arquivo é aberto para a escrita
FILE_BIN modo binário de leitura e escrita(sem conversão de uma string e para uma string)
FILE_CSV arquivo do tipo csv (todos os itens registrados são convertidos para as strings de tipo unicode ou ansi, e são separados por um delimitador)
FILE_TXT um arquivo de texto simples (o mesmo que csv, mas o delimitador não é levado em conta)
FILE_ANSI linhas do tipo ANSI (símbolos de um byte)
FILE_UNICODE linhas do tipo UNICODE (caracteres de bytes duplos)
FILE_SHARE_READ leitura partilhada entre vários programas
FILE_SHARE_WRITE escrita partilhada entre vários programas
FILE_COMMON localização do arquivo em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files.

delimiter='\t'

[in] valor a ser usado como um separador em txt ou csv-file. Se o arquivo .csv delimitador não for especificado, o padrão é uma tabulação. Se o arquivo .txt delimitador não for especificado, então nenhum separador é usado. Se o separador está claramente definido para 0, então nenhum separador é usado.

codepage=CP_ACP

[in] O valor do código de página. Para os mais usados códigos de página forneça as constantes apropriadas.

Valor do Retorno

Se um arquivo foi aberto com sucesso, a função retorna o manipulador de arquivo, que é então usado para acessar os dados do arquivo. Em caso de falha retorna INVALID_HANDLE.

Observação

© 2000-2024, MetaQuotes Ltd.

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Se o arquivo precisar ser lido em uma codificação específica (o parâmetro codepage é especificado com o valor da página de códigos), deverá ser definido o sinalizador FILE_ANSI. Se o sinalizador FILE_ANSI não for especificado, o arquivo de texto será lido em Unicode, sem nenhuma conversão.

O arquivo é aberto na pasta do terminal do cliente na subpasta MQL5\files (ou testing_agent_directory\MQL5\files em caso de testes). Se FILE_COMMON é especificado entre bandeiras, o arquivo é aberto em uma pasta compartilhada para todos os MetaTrader 5 terminais de cliente.

"Pipes nomeados" pode ser aberta de acordo com as seguintes regras:

- Nome do pipe é uma string, que deve ter o seguinte aspecto: "\\servername\pipe\pipeName", onde servername - nome do servidor na rede, enquanto pipename é um nome de tubo. Se os tubos são utilizados no mesmo computador, o nome do servidor pode ser omitido, mas num ponto devem ser inseridos em vez do mesmo: "\\.\pipe\pipename". Um cliente tenta conectar o tubo deve saber o seu nome.
- FileFlush() e FileSeek() deve ser chamado para o início de um arquivo entre as operações seqüenciais de leitura do tubo e gravá-lo.

Um símbolo '\' especial é usado em strings expostas. Portanto, '\' deve ser redobrada quando se escreve um nome na aplicação MQL5. Isso significa que o exemplo acima deve ter o seguinte aspecto no código: "\\\\servername\\pipe\\pipename".

Mais informações sobre como trabalhar com pipes nomeados pode ser encontrada no artigo "Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs".

Exemplo:

```
//+------------------------------------------------------------------+
//| Programa Script da funзro start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- мйtodo de abertura de arquivo incorreto
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("Falha para abrir o arquivo pelo caminho absoluto ");
      Print("Cydigo de erro ",GetLastError());
     }
//--- maneira correta de trabalhar no "sandbox arquivo"
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("FileOpen OK");
     }
   else Print("Operaзro FileOpen falhou, erro ",GetLastError());
//--- outro exemplo com a criaзro de um diretyrio fechado em MQL5\Files\
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("O arquivo serб criado na pasta "+terminal_data_path+"\\"+subfolder);
     }
   else Print("Falha ao abrir arquivo, erro ",GetLastError());
  }
```

Também Veja

Utilização de uma página de código (Codepage), FileFindFirst, FolderCreate, Bandeiras de abertura de arquivos

© 2000-2024, MetaQuotes Ltd.

FileClose

Feche o arquivo previamente aberto por FileOpen().

```
void  FileClose(
   int  file_handle      // Manipulador de arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Sem valor retornado.

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="file.txt";     // nome do arquivo
input string InpDirectoryName="Data";    // nome do diretório
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Imprimir o caminho para o arquivo que vamos usar
   PrintFormat("Trabalhando %s\\Arquivos\\ pasta",TerminalInfoString(TERMINAL_DATA_PAT
//--- redefine o valor de erro
   ResetLastError();
//--- abrir um arquivo para leitura(se não existe arquivo, ocorrerá um erro)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
     {
      //--- imprimir o conteúdo do arquivo
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Erro, código = %d",GetLastError());
  }
```

© 2000-2024, MetaQuotes Ltd.

FileCopy

A função copia o arquivo original a partir de uma pasta local ou compartilhada para outro arquivo.

```
bool  FileCopy(
   const string  src_file_name,      // Nome de um arquivo fonte
   int           common_flag,        // Localização
   const string  dst_file_name,      // Nome do arquivo de destino
   int           mode_flags          // Modo de acesso
   );
```

Parâmetros

src_file_name

[in] O nome do arquivo para cópia.

common_flag

[in] Bandeira determinando a localização do arquivo. Se common_flag = FILE_COMMON, então o arquivo está localizado em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files. Caso contrário, o arquivo está localizado em uma pasta local (por exemplo, *common_flag=0*).

dst_file_name

[in] Resultado nome do arquivo.

mode_flags

[in] Bandeiras de acesso. O parâmetro pode conter apenas duas bandeiras: FILE_REWRITE e/ou FILE_COMMON - outras bandeiras são ignoradas. Se o arquivo já existe, e a bandeira FILE_REWRITE não foi especificado, então o arquivo não será reescrito e a função retornará false.

Valor do Retorno

Em caso de falha a função retornará false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Se o novo arquivo já existe, a cópia será feita dependendo da disponibilidade da bandeira FILE_REWRITE dentro do parâmetro mode_flags.

Exemplo:

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input string InpSrc="source.txt";        // source
input string InpDst="destination.txt";   // copy
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- exibir o conteúdo de origem (que deve existir)
   if(!FileDisplay(InpSrc))
      return;
//--- verifique se a cópia do arquivo já existe (não pode ser criado)
   if(!FileDisplay(InpDst))
     {
      //--- a cópia de arquivo não existe, copiando sem a bandeira FILE_REWRITE (cópia
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Arquivo está copiado!");
      else
         Print("Arquivo não está copiado!");
     }
   else
     {
      //--- cópia do arquivo já existe, tente copiar sem a bandeira FILE_REWRITE (cóp
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Arquivo está copiado!");
      else
         Print("Arquivo não está copiado!");
      //--- conteúdo do arquivo InpDst continua o mesmo
      FileDisplay(InpDst);
      //--- copiar mais uma vez com a bandeira FILE_REWRITE (copiando correto, se o a
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("Arquivo está copiado!");
      else
         Print("Arquivo não está copiado!");
     }
//--- receber cópia de arquivo InpS
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| Ler comentários do arquivo                                       |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- redefine o valor de erro
   ResetLastError();
//--- abre o arquivo
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- exibir o conteúdo do arquivo no loop
      Print("+----------------------+");
      PrintFormat("Nome do arquivo = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
```

© 2000-2024, MetaQuotes Ltd.

```
      Print("+----------------------+");
      //--- fechar o arquivo
      FileClose(file_handle);
      return(true);
     }
//--- falha para abrir o arquivo
   PrintFormat("%s não está aberto, erro = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileDelete

Deletar o arquivo especificado dentro da pasta local do terminal do cliente.

```
bool  FileDelete(
   const string  file_name,      // Nome do arquivo para deletar
   int           common_flag=0   // Localização do arquivo para deletar
   );
```

Parâmetros

file_name

[in] Nome do arquivo.

common_flag=0

[in] Bandeira determinando a localização do arquivo. Se common_flag = FILE_COMMON, então o arquivo está localizado em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files. Caso contrário, o arquivo está localizado em uma pasta local.

Valor do Retorno

Em caso de falha a função retornará false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Exclui o arquivo especificado a partir de uma pasta local do terminal do cliente (MQL5\Files ou MQL5\Tester\Files em caso de testes). Se common_flag = FILE_COMMON, então a função remove o arquivo da pasta compartilhada para todos os terminais do cliente.

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- data para arquivos antigos
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // variável para armazenar os nomes dos arquivos
   string   filter="*.txt"; // filtrar parar pesquisar os arquivos
   datetime create_date;    // data de criação do arquivo
   string   files[];        // lista de nome dos arquivos
   int      def_size=25;    // tamanho do array por padrão
   int      size=0;         // número de arquivos
//--- alocar memória para o array
   ArrayResize(files,def_size);
//--- receber o manipulador de pesquisa na raiz da pasta local
```

© 2000-2024, MetaQuotes Ltd.

```
   long search_handle=FileFindFirst(filter,file_name);
//--- verificar se FileFindFirst() é executada com sucesso
   if(search_handle!=INVALID_HANDLE)
     {
      //--- pesquisando arquivos no loop
      do
        {
         files[size]=file_name;
         //--- aumentar o tamanho do array
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- redefinir o valor de erro
         ResetLastError();
         //--- receber a data de criação do arquivo
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- verificar se o arquivo é antigo
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s arquivo deletado!",file_name);
            //--- deletado o arquivo antigo
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- fechar o manipulador de pesquisa
      FileFindClose(search_handle);
     }
   else
     {
      Print("Arquivo não encontrado!");
      return;
     }
//--- verificar quais os arquivos permanecem
   PrintFormat("Resultados:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s arquivo existe!",files[i]);
      else
         PrintFormat("%s arquivo deletado!",files[i]);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

FileMove

Move um arquivo a partir de um local ou a partir de pasta compartilhada para outra pasta.

```
bool  FileMove(
   const string  src_file_name,     // O nome do arquivo para a operação de movimentaçã
   int           common_flag,       // Locação
   const string  dst_file_name,     // Nome do arquivo de destino
   int           mode_flags         // Modo de acesso
   );
```

Parâmetros

src_file_name

[in] O nome do arquivo para mover/renomear.

common_flag

[in] Bandeira determinando a localização do arquivo. Se common_flag = FILE_COMMON, então o arquivo está localizado em uma pasta compartilhada para todos os terminais de cliente \Terminal\Common\Files. Caso contrário, o arquivo está localizado em uma pasta local (*common_flag=0*).

dst_file_name

[in] O nome do arquivo após a operação

mode_flags

[in] Bandeiras de acesso. O parâmetro pode conter apenas duas bandeiras: FILE_REWRITE e/ou FILE_COMMON - outras bandeiras são ignoradas. Se o arquivo já existe e a bandeira FILE_REWRITE não for especificada, o arquivo não será reescrito, e a função retornará false.

Valor do Retorno

Em caso de falha a função retornará false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Se o novo arquivo já existe, a cópia será feita dependendo da disponibilidade da bandeira FILE_REWRITE dentro do parâmetro mode_flags.

Exemplo:

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Programa Script da função start (iniciar)                          |
//+------------------------------------------------------------------+
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- receber caminhos de arquivo
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- verifica se o arquivo de origem existe (se não - saída)
   if(FileIsExist(src_path))
      PrintFormat("%s file arquivo existe no %s\\Files\\%s pasta",InpSrcName,local,Inp
   else
     {
      PrintFormat("Erro, %s arquivo de origem não encontrado",InpSrcName);
      return;
     }
//--- verifica se o arquivo do resultado já existe
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s arquivo existe no %s\\Files\\%s pasta",InpDstName,common,InpDstI
      //--- arquivo existe, movendo-se deve ser realizado com a flag FILE_REWRITE
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s arquivo movido",InpSrcName);
      else
         PrintFormat("Erro! Código = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s arquivo não existe no %s\\Files\\%s pasta",InpDstName,common,Inp
      //--- arquivo não existe, movendo-se deve ser realizado sem a flag FILE_REWRITE
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s arquivo movido",InpSrcName);
      else
         PrintFormat("Erro! Código = %d",GetLastError());
     }
//--- o arquivo é movido, vamos verificar-lo fora
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Sucesso!");
   else
      Print("Erro!");
  }
```

Também Veja

© 2000-2024, MetaQuotes Ltd.

FileIsExist

© 2000-2024, MetaQuotes Ltd.

FileFlush

Escreve para um disco todos os dados remanescentes dentro da entrada/saída do arquivo buffer.

```
void  FileFlush(
   int  file_handle      // Manipulador de arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Sem valor retornado.

Observação

Ao escrever um arquivo, os dados podem ser realmente encontrados lá só depois de algum tempo. Para salvar os dados no arquivo instantaneamente, utilize a função FileFlush(). Se a função não é utilizada, parte dos dados que não tenham sido ainda armazenados no disco, serão forçosamente escritos lá apenas quando o arquivo é fechado com a funçãoFileClose().

A função deve ser usada quando os dados gravados são de um certo valor. Deve-se ter em mente que freqüente chamada de função podem afetar a velocidade de operação do programa.

Função FileFlush() deve ser chamado entre as operações de leitura de um arquivo e gravá-lo.

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- Nome do arquivo para escrever
input string InpFileName="example.csv"; // nome do arquivo
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- redefine o valor de erro
   ResetLastError();
//--- abre o arquivo
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- gravar dados no arquivo
      for(int i=0;i<1000;i++)
        {
         //--- chamar função de escrever
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sym
         //--- salvar os dados no disco a cada iteração (repetição) 128
         if((i & 127)==127)
```

© 2000-2024, MetaQuotes Ltd.

```
            {
             //--- agora, os dados serão localizados no arquivo e não serão perdidos no
             FileFlush(file_handle);
             PrintFormat("i = %d, OK",i);
            }
          //--- 0.01 segunda pausa
          Sleep(10);
         }
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Erro, código = %d",GetLastError());
  }
```

Também Veja

FileClose

© 2000-2024, MetaQuotes Ltd.

FileGetInteger

Obtém uma propriedade inteira de um arquivo. Existem duas variantes da função.

1. Obtém a propriedade pelo manipulador de um arquivo.

```
long  FileGetInteger(
   int                          file_handle,    // Manipular arquivo
   ENUM_FILE_PROPERTY_INTEGER   property_id     // ID Propriedade
   );
```

2. Obtém a propriedade pelo nome do arquivo.

```
long  FileGetInteger(
   const string                 file_name,            // Nome do arquivo
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // ID Propriedade
   bool                         common_folder=false   // O arquivo é visto em uma pasta
   );                                                 // ou uma pasta comum de todos os
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

file_name

[in] Nome do arquivo.

property_id

[in] Propriedade ID do arquivo. O valor pode ser um dos valores da enumeração ENUM_FILE_PROPERTY_INTEGER. Se a segunda variante da função é usada, você pode receber apenas os valores das seguintes propriedades: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE e FILE_SIZE.

common_folder=false

[in] Pontos para localização de arquivo. Se o parâmetro é false, pasta de dados do terminal é visualizada. Caso contrário, presume-se que o arquivo está na pasta compartilhada por todos os terminais \Terminal\Common\Files(FILE_COMMON).

Valor do Retorno

O valor da propriedade. Em caso de erro, -1 é retornado. Para obter um código de erro usar a função GetLastError().

Se uma pasta está especificada quando obtida a propriedade pelo nome, a função terá erro 5018 (ERR_MQL_FILE_IS_DIRECTORY) em qualquer caso, embora o valor de retorno será correto.

Observação

A função sempre altera o código de erro. Em caso de conclusão bem sucedida, o código de erro é redefinido para NULL.

Exemplo:

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
```

© 2000-2024, MetaQuotes Ltd.

```
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- abre o arquivo
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- imprimir todas as informações sobre o arquivo
      Print(InpFileName," informação de arquivo:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"data");
      FileInfo(handle,FILE_MODIFY_DATE,l,"data");
      FileInfo(handle,FILE_ACCESS_DATE,l,"data");
      FileInfo(handle,FILE_SIZE,l,"outros");
      FileInfo(handle,FILE_POSITION,l,"outros");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- fechar o arquivo
      FileClose(handle);
     }
   else
      PrintFormat("%s arquivo não está aberto, ErrorCode = %d",InpFileName,GetLastErro
  }
//+------------------------------------------------------------------+
//| Mostrar o valor da propriedade de arquivo                        |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- receber o valor da propriedade
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- o valor recebido, exibi-lo no formato correto
```

© 2000-2024, MetaQuotes Ltd.

```
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

Também Veja

Operações de Arquivos, Propriedades de Arquivos

© 2000-2024, MetaQuotes Ltd.

FileIsEnding

Define o final do arquivo em processo de leitura.

```
bool  FileIsEnding(
   int  file_handle      // Manipulador de arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

A função retorna verdadeiro se o fim do arquivo foi alcançado no processo de leitura ou em movimento do ponteiro de arquivo.

Observação

Para definir o final do arquivo, a função tenta ler a próxima string a partir dele. Se a string não existe, a função retorna true, caso contrário, retorna false.

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="file.txt";     // nome do arquivo
input string InpDirectoryName="Data";    // nome do diretório
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Imprimir o caminho para o arquivo que vamos usar
   PrintFormat("Trabalhando %s\\Arquivos\\ pasta",TerminalInfoString(TERMINAL_DATA_PAT
//--- redefine o valor de erro
   ResetLastError();
//--- abrir um arquivo para leitura(se não existe arquivo, ocorrerá um erro)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
     {
      //--- imprimir o conteúdo do arquivo
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
      PrintFormat("Erro, código = %d",GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

FileIsLineEnding

Define o final da linha do arquivo de texto em processo de leitura.

```
bool  FileIsLineEnding(
   int  file_handle      // Manipulador de arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Retorna true se no processo de leitura do arquivo txt ou csv chegou ao fim da linha (os caracteres CR-LF).

Exemplo (o arquivo obtido durante a execução de um exemplo para a função FileWriteString é usado aqui)

```
//+------------------------------------------------------------------+
//|                                        Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Sobre-compra & Sobre-venda"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- paramêtros para leitura de dados
input string InpFileName="RSI.csv";   // nome do arquivo
input string InpDirectoryName="Data"; // nome do diretório
//--- buffers do indicador
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variáveis sobre-comprado
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

© 2000-2024, MetaQuotes Ltd.

```
//--- variáveis sobre-vendido
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- variáveis de tamanhos de array por padrão
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- alocar memória para arrays
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      double value;
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- ler o primeiro valor na string
         value=FileReadNumber(file_handle);
         //--- ler diferentes arrays de acordo com o resultado da função
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- ligando as arrays
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
```

© 2000-2024, MetaQuotes Ltd.

```
//---- definir os valores dos indicadores que não serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Ler dados do arquivo da string                                   |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- é alcançada a leitura até o final da string ou do arquivo
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- deslocar o transporte após a leitura do número
      if(flag)
         FileReadNumber(file_handle);
      //--- armazenar a data atual
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- aumentar tamanho da array, se necessário
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- passar despercebido a primeira iteração
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
```

© 2000-2024, MetaQuotes Ltd.

```
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- verificar se qualquer data continua presente
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- se as datas coincidem, a barra está na zona de sobre-compra
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - cor vermelha
               color_buff[i]=0;
               //--- aumentar o contador
               ovb_ind=j+1;
               break;
              }
           }
      //--- verificar se todos os dados continuam a existir
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- se as datas coincidem, a barra está na zona de sobre-venda
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - cor azul
               color_buff[i]=1;
               //--- aumentar o contador
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Manipulador de eventos ChartEvent                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//---alterar a largura indicador de acordo com a escala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Também Veja

FileWriteString

© 2000-2024, MetaQuotes Ltd.

FileReadArray

Lê a partir de um arquivo do tipo arrays BIN de qualquer tipo, exceto string (pode ser uma array de estruturas, não contendo strings e arrays dinâmicas).

```
uint  FileReadArray(
   int     file_handle,                // Manipular arquivo
   void&   array[],                    // Array para gravar
   int     start=0,                    // Iniciar posição array para escrever
   int     count=WHOLE_ARRAY           // contar para ler
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

array[]

[out] Uma array onde os dados serão carregados.

start=0

[in] Posição inicial para escrever para o array.

count=WHOLE_ARRAY

[in] Número de elementos para ler. Pelo padrão, ler todo o array (count=WHOLE_ARRAY).

Valor do Retorno

Número de elementos para ler.

Observação

Array de seqüências pode ser lido apenas a partir do arquivo do tipo TXT. Se necessário, a função tenta aumentar o tamanho do array.

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWriteArray é usado aqui)

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Estrutura para armazenamento de dados de preços                  |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // data
   double            bid;  // bid preço
   double            ask;  // ask preço
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- estrutura array
   prices arr[];
//--- caminho do arquivo
   string path=InpDirectoryName+"//"+InpFileName;
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- ler todos os dados do arquivo para o array
      FileReadArray(file_handle,arr);
      //--- receber o tamanho do array
      int size=ArraySize(arr);
      //--- imprimir dados a partir do array
      for(int i=0;i<size;i++)
         Print("Data = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Dados total = ",size);
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
      Print("Abrir arquivo falhou, erro ",GetLastError());
  }
```

Também Veja

Variáveis, FileWriteArray

© 2000-2024, MetaQuotes Ltd.

FileReadBool

Lê a partir do arquivo de CSV de tipo string a partir da posição atual para um delimitador (ou até o fim da linha de texto) e converte a leitura string para um valor de tipo bool.

```
bool  FileReadBool(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Linha de leitura pode ser definida como "true", "false" ou a representação simbólica de inteiros "0" ou "1". Um valor diferente de zero é convertido para um true lógico. A função retorna o valor convertido.

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWrite é usado aqui)

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- Plotar Etiqueta1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plotar Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- paramêtros para leitura de dados
input string InpFileName="MACD.csv";  // file name
input string InpDirectoryName="Data"; // nome do diretório
//--- variáveis globais
int      ind=0;       // índice
double   upbuff[];    // buffers indicador de setas para cima
```

© 2000-2024, MetaQuotes Ltd.

```
double   downbuff[];  // buffers indicador de setas para baix
bool     sign_buff[]; // sinal array (verdadeiro - comprar, falso - vender)
datetime time_buff[]; // conjunto de chegada dos sinais time
int      size=0;      // tamanho de arrays de sinal
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo esta aberto para leitura",InpFileName);
      //--- em primeiro lugar, ler o número de sinais
      size=(int)FileReadNumber(file_handle);
      //--- alocar memória para os arrays
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- ler dados de um arquivo
      for(int i=0;i<size;i++)
        {
         //--- tempo de sinal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valor do sinal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- ligando as arrays
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- definir o código símbolo para desenho em PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- definir os valores dos indicadores que não serão vistos no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      upbuff[i]=0;
      downbuff[i]=0;
      //---verificar se todos os dados ainda estão presente
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==time_buff[j])
              {
               //--- desenhar a seta de acordo com o sinal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- aumentar o contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Tipo bool, FileWrite

© 2000-2024, MetaQuotes Ltd.

FileReadDatetime

Lê a partir do arquivo do tipo CSV uma string de um dos formatos: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" ou "HH:MM:SS" - e converte-os dentro de valor do tipo datetime.

```
datetime  FileReadDatetime(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

O valor do tipo datetime.

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWrite é usado aqui)

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- Plotar Etiqueta1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plotar Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- paramêtros para leitura de dados
input string InpFileName="MACD.csv";  // file name
input string InpDirectoryName="Data"; // nome do diretório
//--- variáveis globais
int      ind=0;       // índice
double   upbuff[];    // buffers indicador de setas para cima
double   downbuff[];  // buffers indicador de setas para baix
bool     sign_buff[]; // sinal array (verdadeiro - comprar, falso - vender)
```

© 2000-2024, MetaQuotes Ltd.

```
datetime time_buff[]; // conjunto de chegada dos sinais time
int      size=0;      // tamanho de arrays de sinal
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo esta aberto para leitura",InpFileName);
      //--- em primeiro lugar, ler o número de sinais
      size=(int)FileReadNumber(file_handle);
      //--- alocar memória para os arrays
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- ler dados de um arquivo
      for(int i=0;i<size;i++)
        {
         //--- tempo de sinal
         time_buff[i]=FileReadDatetime(file_handle);
         //--- valor do sinal
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- ligando as arrays
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- definir o código símbolo para desenho em PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- definir os valores dos indicadores que não serão vistos no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      upbuff[i]=0;
      downbuff[i]=0;
      //---verificar se todos os dados ainda estão presente
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==time_buff[j])
              {
               //--- desenhar a seta de acordo com o sinal
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- aumentar o contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Tipo datetime, StringToTime, TimeToString, FileWrite

© 2000-2024, MetaQuotes Ltd.

FileReadDouble

Lê uma dupla precisão de número de ponto (double) flutuante a partir da posição atual do arquivo binário.

```
double  FileReadDouble(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Valor de tipo double.

Observação

Para mais detalhes sobre erro, chamar GetLastError().

Exemplo (o arquivo obtido após a execução do exemplo para a função FileWriteDouble é usado aqui)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- leitura de dados de parâmetros
input string InpFileName="MA.csv";    // nome do arquivo
input string InpDirectoryName="Data"; // nome do diretório
//--- variáveis globais
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- buffer do indicador
double   buff[];
```

```
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //--- em primeiro lugar, ler a quantidade de dados no arquivo
      size=(int)FileReadDouble(file_handle);
      //--- alocar memória para os arrays
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- ler dados de um arquivo
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- Associa o array ao buffer do indicador com índice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- definir os valores dos indicadores que não serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      buff[i]=0;
      //--- verificar se todos os dados continuam a existir
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- aumentar o contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Tipos reais (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2024, MetaQuotes Ltd.

FileReadFloat

Lê a precisão simples de um número (float) de ponto a partir da posição atual do arquivo binário.

```
float  FileReadFloat(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

O valor do tipo float.

Observação

Para mais detalhes sobre erro, chamar GetLastError().

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWriteFloat é usado aqui)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parâmetros para leitura de dados
input string InpFileName="Close.bin"; // file name
input string InpDirectoryName="Data"; // nome do diretório
//--- variáveis globais
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- buffers do indicador
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de inicialização do indicador customizado                  |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- alocar memória para os arrays
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- ler valores de tempo e preço
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- aumentar o tamanho do array, se eles estão transbordados
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Dados são lidos, %s arquivo está fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- ligar arrays para os buffers de indicadores
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- definir os valores dos indicadores que não serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                       |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      buff[i]=0;
      color_buff[i]=0; // cor vermelha por padrão
      //---verificar se todos os dados ainda estão presente
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==time_buff[j])
              {
               //--- receber o preço
               buff[i]=close_buff[j];
               //--- se o preço atual excede o anterior, a cor é azul
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- aumentar o contador
               ind=j+1;
               break;
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Tipos Reais (double, float), FileReadDouble, FileWriteFloat

© 2000-2024, MetaQuotes Ltd.

FileReadInteger

A função lê os valores int, short ou char a partir da posição atual do ponteiro do arquivo, dependendo do comprimento especificado em bytes.

```
int  FileReadInteger(
   int  file_handle,         // Manipular Arquivo
   int  size=INT_VALUE       // Tamanho de um inteiro em bytes
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

size=INT_VALUE

[in] Número de bytes (até 4 inclusive), que devem ser lidos. As constantes correspondentes são fornecidos: CHAR_VALUE = 1, SHORT_VALUE = 2 e INT_VALUE = 4, então a função pode ler todo o valor do tipo char, short ou int.

Valor do Retorno

Um valor do tipo int. O resultado desta função deve ser explicitamente convertido para um tipo de destino, ou seja, para o tipo de dados que você precisa ler. Uma vez que um valor do tipo int é retornado, ele pode ser facilmente convertido para qualquer valor inteiro. O ponteiro do arquivo é deslocado pelo número de bytes lidos.

Observação

Quando ler menos do que 4 bytes, o resultado recebido é sempre positivo. Se um ou dois bytes são lidos, o sinal do número pode ser determinado por colagem explícita do tipo char (1 byte) ou short (2 bytes). Recebendo o sinal para um número de três bytes não é simples, uma vez que não há nenhuma correspondente tipo subjacente.

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWriteInteger é usado aqui)

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_width1  1
//--- parâmetros para leitura de dados
input string InpFileName="Trend.bin"; // file name
input string InpDirectoryName="Data"; // nome do diretório
//--- variáveis globais
int      ind=0;
int      size=0;
datetime time_buff[];
//--- buffers do indicador
double   buff[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- alocar memória para o array
   ArrayResize(time_buff,def_size);
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_
      //--- variáveis adicionais
      int    arr_size;
      uchar  arr[];
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- descobrir quantos símbolos são usados para escrever o tempo
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- armazenar o valor de tempo
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- aumentar as dimensões arrays, se são transbordados
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Dados são lidos, %s arquivo está fechado",InpFileName);
```

```
      }
   else
      {
       PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
       return(INIT_FAILED);
      }
//--- ligar array para buffer de indicador
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- definir os valores dos indicadores que não serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
      {
       //--- 0 por padrão
       buff[i]=0;
       //---verificar se todos os dados ainda estão presente
       if(ind<size)
         {
          for(int j=ind;j<size;j++)
            {
             //--- se as datas coincidem, usar o valor do arquivo
             if(time[i]==time_buff[j])
               {
                //--- receber o preço
                buff[i]=close[i];
                //--- aumentar o contador
                ind=j+1;
                break;
               }
            }
```

```
        }
      }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

IntegerToString, StringToInteger, Tipos inteiros, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileReadLong

A função lê um inteiro do tipo long (8 bytes) a partir da posição atual do arquivo binário.

```
long  FileReadLong(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Valor de tipo long.

Exemplo (o arquivo obtido durante a execução de um exemplo para a função FileWriteLong é usado aqui)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadLong.mq5 |
//|                       Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- parâmetros para leitura de dados
input string InpFileName="Volume.bin"; // file name
input string InpDirectoryName="Data";  // directory name
//--- variáveis globais
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- buffers do indicador
double   buff[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está aberto para escrita",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //--- em primeiro lugar, ler a quantidade de dados no arquivo
      size=(int)FileReadLong(file_handle);
      //--- alocar memória para os arrays
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- ler dados de um arquivo
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Dados são lidos, %s arquivo está fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- associa o array ao buffer do indicador com índice 0
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- definir os valores dos indicadores que serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
```

```
   ArraySetAsSeries(time,false);
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      buff[i]=0;
      //---verificar se todos os dados ainda estão presente
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Tipos Inteiros, FileReadInteger, FileWriteLong

© 2000-2024, MetaQuotes Ltd.

FileReadNumber

A função lê do arquivo CSV uma string a partir da posição atual até um separador (ou até o fim de uma seqüência de texto) e converte a string de leitura para um valor do tipo double.

```
double  FileReadNumber(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Valor de tipo double.

Exemplo (o arquivo obtido durante a execução de um exemplo para a função FileWriteString é usado aqui)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Sobre-compra & Sobre-venda"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- parâmetros para leitura de dados
input string InpFileName="RSI.csv";   // nome do arquivo
input string InpDirectoryName="Data"; // nome do diretório
//--- buffers do indicador
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- variáveis sobre-comprado
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
```

© 2000-2024, MetaQuotes Ltd.

```
//--- variáveis sobre-vendido
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- variáveis de tamanhos de array por padrão
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- alocar memória para arrays
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      double value;
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- ler o primeiro valor na string
         value=FileReadNumber(file_handle);
         //--- ler diferentes arrays de acordo com o resultado da função
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- ligando as arrays
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
```

© 2000-2024, MetaQuotes Ltd.

```
//---- definir os valores dos indicadores que não serão visíveis no gráfico
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Ler dados do arquivo da string                                   |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- é alcançada a leitura até o final da string ou do arquivo
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- deslocar o transporte após a leitura do número
      if(flag)
         FileReadNumber(file_handle);
      //--- armazenar a data atual
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- aumentar tamanho da array, se necessário
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- passar despercebido a primeira iteração
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
```

© 2000-2024, MetaQuotes Ltd.

```
//---o loop para as barras que ainda não foram manipuladas
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- verificar se qualquer data continua presente
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- se as datas coincidem, a barra está na zona de sobre-compra
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - cor vermelha
               color_buff[i]=0;
               //--- aumentar o contador
               ovb_ind=j+1;
               break;
              }
           }
      //--- verificar se todos os dados continuam a existir
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- se as datas coincidem, a barra está na zona de sobre-venda
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - cor azul
               color_buff[i]=1;
               //--- aumentar o contador
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Manipulador de eventos ChartEvent                                |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//---alterar a largura indicador de acordo com a escala
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Também Veja

FileWriteString

© 2000-2024, MetaQuotes Ltd.

FileReadString

A função lê uma string a partir da posição atual do ponteiro de arquivo dentro de um arquivo.

```
string  FileReadString(
   int  file_handle,      // Manipular arquivo
   int  length=-1         // Comprimento string
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

length=-1

[in] Número de caracteres para ler.

Valor do Retorno

Ler linha (string).

Observação

Quando ler a partir de um arquivo bin, o tamanho de uma string para ler deve ser especificada. Quando ler a partir de um arquivo txt, o tamanho da string não é necessária, e a string será lida a partir da posição atual para a alimentação da linha caractere "\r\n". Quando ler a partir de um arquivo CSV,o tamanho da string não é necessária também, a string será lida a partir da posição atual até o mais próximo delimitador ou até string de texto caractere final.

Se o arquivo é aberto com a bandeira FILE_ANSI, então a linha lida é convertida para um Unicode.

Exemplo (o arquivo obtido após a execução do exemplo pela função FileWriteInteger é usado aqui)

```
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- paramêtros para leitura de dados
input string InpFileName="Trend.bin"; // file name
input string InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //--- variáveis adicionais
      int    str_size;
      string str;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- descobrir quantos símbolos são usados para escrever o tempo
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- ler a string
         str=FileReadString(file_handle,str_size);
         //--- imprimir a string
         PrintFormat(str);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Dados são lidos, %s arquivo está fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Tipo string, Conversão de Dados, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileReadStruct

A função lê o conteúdo em uma estrutura passada como um parâmetro a partir de um arquivo binário, começando com a posição atual do ponteiro de arquivo.

```
uint  FileReadStruct(
   int          file_handle,        // Manipular arquivo
   const void&  struct_object,      // estrutura alvo para o qual os conteúdos são lid
   int          size=-1             // tamanho da estrutura em bytes
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo para abrir um arquivo bin.

struct_object

[out] O objeto desta estrutura. A estrutura não deve conter strings, arrays dinâmicas, funções virtuais, objetos de classes, bem como ponteiros para objetos e funções.

size=-1

[in] Número de bytes que devem ser lidos. Se a dimensão não for especificado ou a indicação do valor é maior do que o tamanho da estrutura, o tamanho exato da especificada estrutura é usado.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes lidos. Ponteiro de arquivo é movido pelo mesmo número de bytes.

Exemplo (o arquivo obtido após a utilização do exemplo para a função FileWriteStruct é usado aqui)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- parâmetros para recebimento de dados
input string  InpFileName="EURUSD.txt"; // nome do arquivo
input string  InpDirectoryName="Data";  // nome do diretório
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Estrutura para armazenar dados candlestick                       |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // preço de abertura
   double            close; // preço de fechamento
   double            high;  // preço de máximo
   double            low;   // preço de mínimo
   datetime          date;  // data
  };
//--- buffers do indicador
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- variáveis globais
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- abre o arquivo
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_PATH
      //--- ler dados de um arquivo
      while(!FileIsEnding(file_handle))
        {
         //--- escrever dados para array
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- verificar se array está transbordando
         if(size==default_size)
           {
            //--- aumentar a dimensão array
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- fechar o arquivo
```

```
      FileClose(file_handle);
      PrintFormat("Dados são lidos, %s arquivo está fechado",InpFileName);
     }
   else
     {
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
      return(INIT_FAILED);
     }
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- o loop para os castiçais que ainda não foram manipulados
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 0 por padrão
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //---verificar se todos os dados ainda estão presente
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- se as datas coincidem, usar o valor do arquivo
            if(time[i]==cand_buff[j].date)
              {
```

```
                open_buff[i]=cand_buff[j].open;
                close_buff[i]=cand_buff[j].close;
                high_buff[i]=cand_buff[j].high;
                low_buff[i]=cand_buff[j].low;
                //--- aumentar o contador
                ind=j+1;
                break;
               }
            }
         }
      }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Estruturas e classes, FileWriteStruct

© 2000-2024, MetaQuotes Ltd.

FileSeek

A função move a posição do ponteiro de arquivo por um certo número de bytes de informação em relação à posição especificada.

```
bool  FileSeek(
   int                  file_handle,      // Manipular arquivo
   long                 offset,           // Em bytes
   ENUM_FILE_POSITION   origin            // Posição para referência
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

offset

[in] A mudança de bytes (pode tomar um valor negativo).

origin

[in] O ponto de partida para o deslocamento. Pode ser um dos valores de ENUM_FILE_POSITION.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informação sobre o erro chamar GetLastError().

Observação

Se a execução da função FileSeek() resulta em uma mudança negativa (indo além do "nível limite" do arquivo), o ponteiro do arquivo será definido para o arquivo de início.

Se a posição for definido para além do "limite direito" do arquivo (maior do que o tamanho do arquivo),a próxima escrita para o arquivo será executado não a partir do final do arquivo, mas a partir da posição definida. Neste caso, os valores indefinidos serão escritos para o final do arquivo anterior e a posição definida.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="file.txt";     // nome do arquivo
input string InpDirectoryName="Data";    // nome do diretório
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- especificar o valor da variável para gerar números aleatórios
   _RandomSeed=GetTickCount();
//--- variáveis para posições de pontos de início das strings
   ulong pos[];
   int   size;
//--- redefine o valor de erro
   ResetLastError();
//--- abre o arquivo
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para leitura",InpFileName);
      //--- receber a posição inicial para cada string no arquivo
      GetStringPositions(file_handle,pos);
      //--- definir o número de strings no arquivo
      size=ArraySize(pos);
      if(!size)
        {
         //--- parar se o arquivo não tem strings
         PrintFormat("%s arquivo está vazio!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- fazer uma seleção aleatória de um número de string
      int ind=MathRand()%size;
      //--- deslocar posição de ponto de partida da string
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- ler e imprimir a string com o número ind
         PrintFormat("Texto string com %d número: \"%s\"",ind,FileReadString(file_hand
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("%s arquivo está vazio",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
//+------------------------------------------------------------------------------+
//| A função define pontos de partida para cada uma das strings no arquivo e     |
//| coloca-los em arr array                                                      |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
```

© 2000-2024, MetaQuotes Ltd.

```
//--- array tamanho padrão
   int def_size=127;
//--- alocar memória para o array
   ArrayResize(arr,def_size);
//--- contador string
   int i=0;
//--- se isto não é o fim do arquivo, então há pelo menos uma string
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // o arquivo está vazio, sair
//--- define o deslocamento em bytes, dependendo em codificação
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- ir através das strings no loop
   while(1)
     {
      //--- ler a string
      FileReadString(handle);
      //--- verificar o final do arquivo
      if(!FileIsEnding(handle))
        {
         //--- armazenar a próxima posição da string
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- aumentar o tamanho da array, se ela é transbordada
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // final do arquivo, sair
     }
//--- definir o tamanho real de array
   ArrayResize(arr,i);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileSize

A função retorna o tamanho do arquivo em bytes.

```
ulong  FileSize(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

O valor do tipo int.

Observação

Para obter informação sobre o erro chamar GetLastError().

Exemplo:

```
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros de entrada
input ulong  InpThresholdSize=20;        // tamanho limite do arquivo em kilobytes
input string InpBigFolderName="big";     // pasta para arquivos grandes
input string InpSmallFolderName="small"; // pasta para arquivo pequeno
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // variável para armazenar os nomes dos arquivos
   string   filter="*.csv"; // filtro para pesquisar os arquivos
   ulong    file_size=0;    // tamanho do arquivo em bytes
   int      size=0;         // número de arquivos
//--- imprimir o caminho para o arquivo que vamos trabalhar com
   PrintFormat("Trabalhando em pasta %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDA
//--- receber o manipulador de pesquisa em comum na raiz da pasta de todos os termina
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- verificar se FileFindFirst() é executada com sucesso
   if(search_handle!=INVALID_HANDLE)
     {
      //--- mover arquivos no loop de acordo com seus tamanhos
      do
        {
         //--- abrir o arquivo
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
```

© 2000-2024, MetaQuotes Ltd.

```
            {
             //--- receber o tamanho do arquivo
             file_size=FileSize(file_handle);
             //--- fechar o arquivo
             FileClose(file_handle);
            }
          else
            {
             PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",file_name,(
             continue;
            }
          //--- imprimir o tamanho do arquivo
          PrintFormat("Tamanho do %s arquivo é igual para %d bytes",file_name,file_size
          //--- define o caminho para movimentar o arquivo
          string path;
          if(file_size>InpThresholdSize*1024)
             path=InpBigFolderName+"//"+file_name;
          else
             path=InpSmallFolderName+"//"+file_name;
          //--- mover o arquivo
          ResetLastError();
          if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
             PrintFormat("%s arquivo foi movido",file_name);
          else
             PrintFormat("Erro, código = %d",GetLastError());
         }
      while(FileFindNext(search_handle,file_name));
      //--- fechar o manipulador de pesquisa
      FileFindClose(search_handle);
     }
   else
      Print("Arquivo não encontrado!");
  }
```

© 2000-2024, MetaQuotes Ltd.

FileTell

O arquivo retorna a posição atual do ponteiro de arquivo de um arquivo aberto.

```
ulong  FileTell(
   int  file_handle    // Manipular arquivo
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

Valor do Retorno

Posição atual do descritor de arquivo em bytes a partir do início do arquivo.

Observação

Para obter informação sobre o erro chamar GetLastError().

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mosta a janela dos parâmetros de entrada quando lançado o script
#property script_show_inputs
//--- parâmetros de entrada
input string InpFileName="file.txt";     // nome do arquivo
input string InpDirectoryName="Data";    // nome do diretório
input int    InpEncodingType=FILE_ANSI; // ANSI=32 ou UNICODE=64
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- especificar o valor da variável para gerar números aleatórios
   _RandomSeed=GetTickCount();
//--- variáveis para posições de pontos de início das strings
   ulong pos[];
   int   size;
//--- redefine o valor de erro
   ResetLastError();
//--- abre o arquivo
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       PrintFormat("%s arquivo está disponível para leitura",InpFileName);
       //--- receber a posição inicial para cada string no arquivo
       GetStringPositions(file_handle,pos);
       //--- definir o número de strings no arquivo
       size=ArraySize(pos);
       if(!size)
         {
          //--- parar se o arquivo não tem strings
          PrintFormat("%s arquivo está vazio!",InpFileName);
          FileClose(file_handle);
          return;
         }
       //--- fazer uma seleção aleatória de um número de string
       int ind=MathRand()%size;
       //--- deslocar posição de ponto de partida da string
       FileSeek(file_handle,pos[ind],SEEK_SET);
       //--- ler e imprimir a string com o número ind
       PrintFormat("Texto string com %d número: \"%s\"",ind,FileReadString(file_handle)
       //--- fechar o arquivo
       FileClose(file_handle);
       PrintFormat("%s arquivo está vazio",InpFileName);
      }
    else
       PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
//+------------------------------------------------------------------------------+
//| A função define pontos de partida para cada uma das strings no arquivo e     |
//| coloca-los em arr array                                                      |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- array tamanho padrão
   int def_size=127;
//--- alocar memória para o array
   ArrayResize(arr,def_size);
//--- contador string
   int i=0;
//--- se isto não é o fim do arquivo, então há pelo menos uma string
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // o arquivo está vazio, sair
//--- define o deslocamento em bytes, dependendo em codificação
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
```

© 2000-2024, MetaQuotes Ltd.

```
      shift=1;
   else
      shift=2;
//--- ir através das strings no loop
   while(1)
     {
      //--- ler a string
      FileReadString(handle);
      //--- verificar o final do arquivo
      if(!FileIsEnding(handle))
        {
         //--- armazenar a próxima posição da string
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- aumentar o tamanho da array, se ela é transbordada
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // final do arquivo, sair
     }
//--- definir o tamanho real de array
   ArrayResize(arr,i);
  }
```

© 2000-2024, MetaQuotes Ltd.

FileWrite

A função destina-se a gravação de dados em um arquivo CSV, delimitador sendo inserido automaticamente a menos que seja igual a 0. Depois de escrever dentro do arquivo, o caractere de final de linha "\r\n" será adicionado.

```
uint  FileWrite(
   int  file_handle,    // manipular arquivo
   ...                  // Lista de parâmetros gravados
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

...

[in] A lista de parâmetros separados por vírgulas, para escrever no arquivo. O número de parâmetros escritos pode ser de até 63.

Valor do Retorno

Número de bytes escritos.

Observação

Os números serão convertidos dentro de um texto na saída (veja a print() função). Dados do tipo double são emitidos com a precisão de 16 dígitos depois do ponto decimal, e os dados podem ser exibidos quer no formato tradicional ou no formato científico - dependendo de qual formato será o mais compacto. Os dados do tipo float são mostrados com cinco dígitos depois do ponto decimal. Para emitir números reais com precisão diferente ou em um formato claramente especificado, use DoubleToString().

Números do tipo bool são exibidos como "true" ou "false" strings. Números do tipo datetime são apresentados como "YYYY.MM.DD HH:MI:SS".

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string             InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tempo
input int                InpFastEMAPeriod=12;              // período rápido EMA
input int                InpSlowEMAPeriod=26;              // período lento EMA
```

© 2000-2024, MetaQuotes Ltd.

```
input int                 InpSignalPeriod=9;                // diferença de período méd
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo de preço
input datetime            InpDateStart=D'2012.01.01 00:00'; // data de início da cópia
//--- parâmetros para a gravação de dados ao arquivo
input string              InpFileName="MACD.csv";  // nome do arquivo
input string              InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // copiando dados com a data final
   bool     sign_buff[]; // sinal array (verdadeiro - comprar, falso - vender)
   datetime time_buff[]; // array de chegada de tempo de sinais
   int      sign_size=0; // tamanho de array
   double   macd_buff[]; // array do valor do indicador
   datetime date_buff[]; // array da data do indicador
   int      macd_size=0; // tamanho de arrays de indicadores
//--- tempo final é o tempo corrente
   date_finish=TimeCurrent();
//--- receber a manipulação do indicador MACD
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- falha para receber a manipulação do indicador
      PrintFormat("Erro quando recebe a manipulação do indicador. Código de erro = %d"
      return;
     }
//--- estando no loop até que o indicador calcula todos os seus valores
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // pausa para permitir que o indicador calcule os respectivos valores
//--- copiar os valores do indicador para um determinado período de tempo
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Falha para copiar valores do indicador. Código de erro = %d",GetLas
      return;
     }
//--- copiar o tempo apropriado para os valores do indicador
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastErr
      return;
     }
//--- libera a memória ocupada pelo indicador
   IndicatorRelease(macd_handle);
//--- receber o tamanho do buffer
```

© 2000-2024, MetaQuotes Ltd.

```
   macd_size=ArraySize(macd_buff);
//--- analisar os dados e salva os sinais do indicador para arrays
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- sinal comprar
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- sinal vender
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- abrir o arquivo para escrever os valores do indicador (se o arquivo estiver ause
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para ser escrito",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_
      //--- primeiro, escreva o número de sinais
      FileWrite(file_handle,sign_size);
      //--- escrever o tempo e os valores de sinais para o arquivo
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Comment, Print, StringFormat

© 2000-2024, MetaQuotes Ltd.

FileWriteArray

A função escreve arrays de qualquer tipo, exceto para string de um arquivo bin (pode strings ou arrays dinâmicas).

```
uint  FileWriteArray(
   int          file_handle,         // Manipular arquivo
   const void&  array1[],            // Array
   int          start=0,             // Iniciar índice array
   int          count=WHOLE_ARRAY    // Número de elementos
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

array[]

[out] Array para gravação.

start=0

[in] Início do índice array (número do primeiro elemento gravado).

count=WHOLE_ARRAY

[in] Número de itens para gravar(WHOLE_ARRAY significa que as gravações de todos os itens começam com o número inicial até o final da array).

Valor do Retorno

Número de itens gravados.

Observação

String array pode ser gravada em arquivo TXT. Neste caso, strings são automaticamente terminadas pela linha final de caracteres "\r\n". Dependendo sobre o arquivo tipo ANSI ou UNICODE, strings são ambas convertidas para ansi-codificação ou não.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- parâmetros de entrada
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Estrutura para armazenamento de dados de preços                  |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
struct prices
  {
   datetime          date; // data
   double            bid;  // bid preço
   double            ask;  // ask preço
  };
//--- variáveis globais
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- alocar memória para o array
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- escrever a contagem restante de strings se count < n
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- salvar dados para array
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- mostra dados correntes
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//--- aumentar o contador
   count++;
//--- se array é arquivada, gravar dados no arquivo e zerá-lo
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Escrever n elementos array para arquivo                          |
//+------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- abre o arquivo
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- escrever os dados array para o final do arquivo
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- fechar o arquivo
      FileClose(handle);
     }
   else
      Print("Falha para abrir o arquivo, erro ",GetLastError());
  }
```

Também Veja

Variáveis, FileSeek

© 2000-2024, MetaQuotes Ltd.

FileWriteDouble

A função escreve o valor de um parâmetro duplo para um arquivo bin, a partir da posição atual do ponteiro de arquivo.

```
uint  FileWriteDouble(
   int     file_handle,      // Manipular arquivo
   double  value             // Valor para escrever
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

value

[in] O valor do tipo double.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos (neste caso sizeof(double)=8). O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string             InpSymbolName="EURJPY";           // par de moedas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // período de tempo
input int                InpMAPeriod=10;                   // período suavizado
input int                InpMAShift=0;                     // deslocacamente de indica
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // tipo de suavização
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo de preço
input datetime           InpDateStart=D'2013.01.01 00:00'; // data de início da cópia
//--- parâmetros para escrever dados no arquivo
input string             InpFileName="MA.csv";    // file name
input string             InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2024, MetaQuotes Ltd.

```
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- receber a manipulador do indicador MA
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- falha para receber a manipulação do indicador
      PrintFormat("Erro quando recebe a manipulação do indicador. Código de erro = %d"
      return;
     }
//--- estando no loop até que o indicador calcula todos os seus valores
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // pausa para permitir que o indicador calcule os respectivos valores
   PrintFormat("Valores dos indicadores começando a partir de %s será escrita no arqui
//--- copiar os valores dos indicadores
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Falha para copiar valores do indicador. Código de erro = %d",GetLas
      return;
     }
//--- copiar o tempo da apropriada chegada das barras
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastEr
      return;
     }
//--- receber o tamanho do buffer
   size=ArraySize(ma_buff);
//--- libera a memória ocupada pelo indicador
   IndicatorRelease(ma_handle);
//--- abrir o arquivo para escrever os valores do indicador (se o arquivo estiver ause
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para ser escrito",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_
      //--- em primeiro lugar, escrever o tamanho da amostra de dados
      FileWriteDouble(file_handle,(double)size);
      //--- escrever o tempo do indicador e valor para o arquivo
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Tipos reais (double, float)

© 2000-2024, MetaQuotes Ltd.

FileWriteFloat

A função escreve o valor do parâmetro float para um arquivo bin, a partir da atual posição do ponteiro de arquivo.

```
uint  FileWriteFloat(
   int    file_handle,      // Manipular arquivo
   float  value             // Valor para ser escrito
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

value

[in] O valor do tipo float.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos (neste caso sizeof(float)=4). O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string          InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // período de tempo
input datetime        InpDateStart=D'2013.01.01 00:00'; // data de início da cópia dos
//--- parâmetros para escrever dados no arquivo
input string          InpFileName="Close.bin"; // nome do arquivo
input string          InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redefine o valor de erro
   ResetLastError();
//--- copiar o preço de fechamento de cada barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Falha ao copiar valores de preço de fechamento. Código de erro = %d
      return;
     }
//--- copiar o tempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastErr
      return;
     }
//--- receber o tamanho do buffer
   size=ArraySize(close_buff);
//--- abrir o arquivo para escrever os valores (se o arquivo estiver ausente, ele sera
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está aberto para escrita",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //--- escrever fechamento do tempo dos preços e os valores para o arquivo
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Tipos reais (double, float), FileWriteDouble

© 2000-2024, MetaQuotes Ltd.

FileWriteInteger

A função escreve o valor do parâmetro int para um arquivo bin, a partir da posição atual do ponteiro de arquivo.

```
uint  FileWriteInteger(
   int  file_handle,         // Manipular Arquivo
   int  value,               // Valor para ser escrito
   int  size=INT_VALUE       // Tamanho em bytes
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

value

[in] Valor inteiro.

size=INT_VALUE

[in] Número de bytes (até 4 inclusive), que deve ser escrito. As constantes correspondentes são fornecidos: CHAR_VALUE=1, SHORT_VALUE=2 e INT_VALUE=4, então a função pode escrever o valor inteiro do tipo char, uchar, short, ushort, int, ou uint.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos. O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string             InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tempo
input datetime           InpDateStart=D'2013.01.01 00:00'; // data de início da cópia
//--- parâmetros para escrever dados no arquivo
input string             InpFileName="Trend.bin"; // nome do arquivo
input string             InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- redefine o valor de erro
   ResetLastError();
//--- copiar o preço de fechamento de cada barra
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Falha ao copiar os valores dos preços de fechamento. Código de erro
      return;
     }
//--- copiar o tempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastErr
      return;
     }
//--- receber o tamanho do buffer
   size=ArraySize(close_buff);
//--- abrir o arquivo para escrever os valores (se o arquivo estiver ausente, ele será
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para ser escrito",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_P
      //---
      int   up_down=0; // tendência flag
      int   arr_size;  // tamanho de array arr
      uchar arr[];     // array tipo uchar
      //--- escrever valores de tempo para o arquivo
      for(int i=0;i<size-1;i++)
        {
         //--- comparar preços de fechamento das atuais e próximas barras
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- escrever valor de data para o arquivo usando FileWriteInteger
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- primeiro, escreva o número de símbolos na array
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- escrever os símbolos
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- mudar a flag de tendência
```

© 2000-2024, MetaQuotes Ltd.

```
                up_down=1;
               }
            }
         else
            {
             if(up_down!=-1)
               {
                //--- escrever valor de data para o arquivo usando FileWriteInteger
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- primeiro, escreva o número de símbolos na array
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- escrever os símbolos
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- mudar a flag de tendência
                up_down=-1;
               }
            }
         }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

IntegerToString, StringToInteger, Tipos Inteiros

© 2000-2024, MetaQuotes Ltd.

FileWriteLong

A função escreve o valor do parâmetro de tipo long para um arquivo bin, a partir da posição atual do ponteiro de arquivo.

```
uint  FileWriteLong(
   int   file_handle,      // Manipular arquivo
   long  value             // Valor para ser escrito
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

value

[in] Valor do tipo long.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos (neste caso sizeof(long)=8). O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string          InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // time frame
input datetime        InpDateStart=D'2013.01.01 00:00'; // data de início da cópia dos
//--- parâmetros para escrever dados no arquivo
input string          InpFileName="Volume.bin"; // nome do arquivo
input string          InpDirectoryName="Data";  // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- redefine o valor de erro
   ResetLastError();
//--- copiar volumes de tick para cada bar
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_bu
     {
      PrintFormat("Falha ao copiar valores do volume de tick. Código de erro = %d",Get
      return;
     }
//--- copiar o tempo para cada barra
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastEr
      return;
     }
//--- receber o tamanho do buffer
   size=ArraySize(volume_buff);
//--- abrir o arquivo para escrever os valores do indicador (se o arquivo estiver aus
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para ser escrito",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_
      //--- primeiro, escreva o tamanho da amostra de dados
      FileWriteLong(file_handle,(long)size);
      //--- escrever valores de tempo e de volume para o arquivo
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Tipos inteiros, FileWriteInteger

© 2000-2024, MetaQuotes Ltd.

FileWriteString

A função escreve o valor de um parâmetro do tipo string em um arquivo BIN, CSV ou TXT começando a partir da posição atual do ponteiro de arquivo. Quando escrever para um arquivo CSV ou TXT: se existe um símbolo na string '\n' (LF) sem caractere anterior '\r' (CR), então antes de '\n' desaparecer '\r' é adicionado.

```
uint  FileWriteString(
   int           file_handle,     // Manipular arquivo
   const string  text_string,     // string para escrever
   int           length=-1        // número de símbolos
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

text_string

[in] Seqüência.

length=-1

[in] O número de caracteres que você quer escrever. Esta opção é necessária para escrever uma string em um arquivo BIN. Se o tamanho não for especificado, então a string inteira, sem o trailer 0 (zero) é escrita. Se você especificar um tamanho menor do que o comprimento da string, então uma parte da string sem o trailer 0 (zero) é escrita. Se você especificar um tamanho maior do que o comprimento da string, a string é preenchido por um número apropriado de zeros. Para arquivos do tipo CSV e TXT, este parâmetro é ignorado e a string é escrita inteiramente.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos. O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Observação

Note que quando se escreve para um arquivo aberto pela bandeira FILE_UNICODE (ou sem uma bandeira FILE_ANSI), então o número de bytes escrito será duas vezes tão grande como o número de cadeia de string escritas. Quando gravar para um arquivo aberto com a bandeira FILE_ANSI, o número de bytes escritos irá coincidir com o número de caracteres de string escritas.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
```

```
//--- parâmetros para a recepção de dados a partir do terminal
input string             InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // período de tempo
input int                InpMAPeriod=14;                   // período de média móvel
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // tipo de preço
input datetime           InpDateStart=D'2013.01.01 00:00'; // data de início da cópia
//--- parâmetros para escrever dados no arquivo
input string             InpFileName="RSI.csv";   // nome do arquivo
input string             InpDirectoryName="Data"; // nome do diretório
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // copiando dados com a data final
   double   rsi_buff[];  // array do valor do indicador
   datetime date_buff[]; // array da data do indicador
   int      rsi_size=0;  // tamanho de arrays de indicadores
//--- tempo final é o atual
   date_finish=TimeCurrent();
//--- receber a manipulação do indicador RSI
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- falha para receber a manipulação do indicador
      PrintFormat("Erro quando recebe a manipulação do indicador. Código de erro = %d"
      return;
     }
//--- fica no loop, até que o indicador calcula todos os seus valores
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // pausa para permitir que o indicador calcule os respectivos valores
//--- copiar os valores do indicador para um determinado período de tempo
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Falha para copiar valores do indicador. Código de erro = %d",GetLas
      return;
     }
//--- copiar o tempo apropriado para os valores do indicador
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastEr
      return;
     }
//--- libera a memória ocupada pelo indicador
   IndicatorRelease(rsi_handle);
//--- receber o tamanho do buffer
```

© 2000-2024, MetaQuotes Ltd.

```
   rsi_size=ArraySize(rsi_buff);
//--- abrir o arquivo para escrever os valores do indicador (se o arquivo estiver ause
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está disponível para ser escrito",InpFileName);
      PrintFormat("Caminho do arquivo: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_
      //--- preparar variáveis adicionais
      string str="";
      bool   is_formed=false;
      //--- escrever datas de formação das áreas de sobre-compra e sobre-venda
      for(int i=0;i<rsi_size;i++)
        {
         //--- verifique os valores dos indicadores
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- se o valor é o primeiro nesta área
            if(!is_formed)
              {
               //--- adicionar o valor e a data
               str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
               is_formed=true;
              }
            else
               str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
            //--- mover para a próxima iteração do loop
            continue;
           }
         //--- verificar a flag
         if(is_formed)
           {
            //--- a string é formada, escrevê-lo para o arquivo
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Tipo string, StringFormat

© 2000-2024, MetaQuotes Ltd.

FileWriteStruct

A função escreve em arquivo bin um conteúdo de uma estrutura passada como um parâmetro, começando a partir da posição atual do ponteiro de arquivo.

```
uint  FileWriteStruct(
   int          file_handle,       // Manipular arquivo
   const void&  struct_object,     // link para um objeto
   int          size=-1            // tamanho a ser escrito em bytes
   );
```

Parâmetros

file_handle

[in] Descritor de arquivo retornado pelo FileOpen().

struct_object

[in] Referência ao objeto desta estrutura. A estrutura não deve conter strings, arrays dinâmicas, funções virtuais, objetos de classes, bem como ponteiros para objetos e funções.

size=-1

[in] Número de bytes que você quer gravar. Se tamanho não é especificado ou o especificado número de bytes é maior do que o tamanho da estrutura, a estrutura inteira é escrita.

Valor do Retorno

Se for bem sucedido, a função retorna o número de bytes escritos. O ponteiro do arquivo é deslocado pelo mesmo número de bytes.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- mostrar a janela de parâmetros de entrada quando do lançamento do script
#property script_show_inputs
//--- parâmetros para a recepção de dados a partir do terminal
input string          InpSymbolName="EURUSD";           // par de moedas
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // time frame
input datetime        InpDateStart=D'2013.01.01 00:00'; // data de início da cópia dos
//--- parâmetros para escrever dados no arquivo
input string          InpFileName="EURUSD.txt";         // nome do arquivo
input string          InpDirectoryName="Data";          // nome do diretório
//+------------------------------------------------------------------+
//| Estrutura para armazenar dados candlestick                       |
//+------------------------------------------------------------------+
struct candlesticks
```

© 2000-2024, MetaQuotes Ltd.

```
  {
   double            open;  // preço de abertura
   double            close; // preço de fechamento
   double            high;  // preço de máximo
   double            low;   // preço de mínimo
   datetime          date;  // data
  };
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- redefine o valor de erro
   ResetLastError();
//--- receber o tempo da chegada das barras a partir do intervalo
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Falha para copiar valores de tempo. Código de erro = %d",GetLastEr
      return;
     }
//--- receber os preços de máximo das barras a partir do intervalo
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("Falha ao copiar os valores dos preços de máximo. Código de erro = %
      return;
     }
//--- receber os preços de mínimo das barras a partir do intervalo
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Falha ao copiar os valores dos preços de mínimo. Código de erro = %
      return;
     }
//--- receber os preços de abertura das barras a partir do intervalo
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("Falha ao copiar os valores dos preços de abertura. Código de erro =
      return;
     }
//--- receber os preços de fechamento das barras a partir do intervalo
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
```

```
      PrintFormat("Falha ao copiar os valores dos preços de fechamento. Código de erro
      return;
     }
//--- definir o tamanho de arrays
   size=ArraySize(time_buff);
//--- salvar todos os dados na estrutura array
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- abrir o arquivo para escrever a estrutura array para o arquivo (se o arquivo est
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FIL
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s arquivo está aberto para escrita",InpFileName);
      PrintFormat("File path: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_PATH
      //--- preparar o contador do número de bytes
      uint counter=0;
      //--- escrever valores array no loop
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("%d bytes de informação está escrito para %s arquivo",InpFileName,co
      PrintFormat("Número total de bytes: %d * %d * %d = %d, %s",size,5,8,size*5*8,siz
      //--- fechar o arquivo
      FileClose(file_handle);
      PrintFormat("Os dados são escritos, %s arquivo esta fechado",InpFileName);
     }
   else
      PrintFormat("Falha para abrir %s arquivo, Código de erro = %d",InpFileName,GetLa
  }
```

Também Veja

Estruturas e classes

© 2000-2024, MetaQuotes Ltd.

FileLoad

Ele lê todo o conteúdo do arquivo binário especificado numa variedade transferida de tipos numéricos ou estruturas simples. A função permite ler rapidamente os dados de um tipo conhecido na matriz apropriada.

```
long  FileLoad(
   const string  file_name,          // nome do arquivo
   void&         buffer[],           // matriz de tipos numéricos ou estruturas simples
   int           common_flag=0       // sinalizador de arquivo, por padrão o arquivo é
   );
```

Parâmetros

file_name

[in] Nome do arquivo a partir do qual serão lidos os dados.

buffer

[out] Matriz de tipos numéricos ou estruturas simples.

common_flag=0

[in] Sinalizador de arquivo que indica o modo de trabalho. Se este parâmetro não for especificado, o arquivo é procurado na subpasta MQL5\Files (ou <diretório_do_agente_de_teste>\MQL5\Files em caso de teste).

Valor de retorno

Número de elementos lidos ou -1 em caso de falha.

Observação

A função FileLoad() lê, a partir do arquivo, o número de bytes múltiplo do tamanho da matriz. Por exemplo, suponha que o tamanho do arquivo é 10 bytes e a leitura de dados é realizada numa matriz do tipo double (sizeof(double)=8). Neste caso, você vai ler apenas 8 bytes, os restantes 2 bytes no final do arquivo serão simplesmente descartados e a função FileLoad() retornará 1 (lido 1 elemento).

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_FileLoad.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input int      bars_to_save=10; // número de barras
```

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) copied %d bars",_Symbol,copied);
      //---  escrevemos as cotações no arquivo
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() failed, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
//--- agora lemos estas cotações de volta para o arquivo
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

Veja também

Estruturas e classes, FileReadArray, FileReadStruct, FileSave

© 2000-2024, MetaQuotes Ltd.

FileSave

Armazena no arquivo binário todos os elementos da matriz transferida como parâmetro. A função permite armazenar rapidamente -numa cadeia de caracteres- matrizes de tipos numéricos ou estruturas simples.

```
bool  FileSave(
   const string  file_name,          // nome do arquivo
   void&         buffer[],           // matriz de tipos numéricos ou estruturas simples
   int           common_flag=0       // sinalizador de arquivo, por padrão os arquivos
   );
```

Parâmetros

file_name

[in] Nome do arquivo no qual será armazenada a matriz de dados.

buffer

[in] Matriz de tipos numéricos ou estruturas simples.

common_flag=0

[in] Sinalizador de arquivo que indica o modo de trabalho. Se este parâmetro não for especificado, o arquivo será armazenado na subpasta MQL5\Files (ou <diretório_do_agente_de_teste>\MQL5\Files em caso de teste).

Valor de retorno

Em caso de falha, a função retorna false.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input int      ticks_to_save=1000; // número de ticks
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
```

```
      {
       PrintFormat(" CopyTicks(%s) copied %d ticks",_Symbol,copied);
       //--- se o histórico de ticks estiver sincronizada, o código de erro será igual
       if(!GetLastError()==0)
          PrintFormat("%s: Ticks are not synchronized, error=%d",_Symbol,copied,_LastEr
       //---  armazenamos os ticks no arquivo
       if(!FileSave(filename,ticks,FILE_COMMON))
          PrintFormat("FileSave() failed, error=%d",GetLastError());
      }
    else
       PrintFormat("Failed CopyTicks(%s), Error=",_Symbol,GetLastError());
//--- agora lemos estes ticks de volta para o arquivo
    ArrayFree(ticks);
    long count=FileLoad(filename,ticks,FILE_COMMON);
    if(count!=-1)
      {
       Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
       for(int i=0;i<count;i++)
         {
          PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
          TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
          ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
         }
      }
   }
```

Veja também

Estruturas e classes, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

© 2000-2024, MetaQuotes Ltd.

FolderCreate

A função cria uma pasta no diretório Arquivos (dependendo do valor de common_flag).

```
bool  FolderCreate(
   string  folder_name,        // String com o nome da nova pasta
   int     common_flag=0       // escopo
   );
```

Parâmetros

folder_name

[in] O nome do diretório que deseja criar. Contém o caminho relativo para a pasta.

common_flag=0

[in] Bandeira determinando a localização do diretório. Se common_flag=FILE_COMMON, então o diretório está na pasta compartilhada para todos os terminais do cliente \Terminal\Common\Files. Caso contrário, o diretório está em uma pasta local (MQL5\Files ou MQL5\Tester\Files na caso de testes).

Valor do Retorno

Retorna true se bem-sucedida, caso contrário - false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Exemplo:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- descrição
#property description "O script mostra um exemplo de utilização da FolderCreate()."
#property description "Parâmetro externo define a pasta para criação de pastas."
#property description "Após a execução do script, será criada a estrutura das pastas"

//--- mostramos a janela de parâmetros de entrada ao executar o script
#property script_show_inputs
//--- o parâmetro de entrada define a pasta em que está sendo executado o script
input bool     common_folder=false; // pasta compartilhada de todos os terminais
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- pasta que criamos em MQL5\Files
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- nela, criamos uma pasta filha Child_Folder_B1
       string folder_B1="Child_Folder_B1";
       string path=root_folder+"\\"+folder_B1;          // criamos o nome da pasta com
       if(CreateFolder(path,common_folder))
         {
          //--- nesta pasta, criamos mais 3 filhas
          string folder_C11="Child_Folder_C11";
          string child_path=root_folder+"\\"+folder_C11;// criamos o nome da pasta com
          CreateFolder(child_path,common_folder);
          //--- segunda pasta filha
          string folder_C12="Child_Folder_C12";
          child_path=root_folder+"\\"+folder_C12;
          CreateFolder(child_path,common_folder);

          //--- terceira pasta filha
          string folder_C13="Child_Folder_C13";
          child_path=root_folder+"\\"+folder_C13;
          CreateFolder(child_path,common_folder);
         }
      }
//---
  }
//+------------------------------------------------------------------+
//| Tenta criar a pasta e exibe uma mensagem sobre isto              |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- definimos o caminho completo dependendo do parâmetro common_flag
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- mensagem de depuração
   PrintFormat("folder_path=%s",folder_path);
//--- tentativa de criar a pasta no caminho relativo MQL5\Files
   if(FolderCreate(folder_path,flag))
     {
      //--- exibimos o caminho completo para a pasta criada
      PrintFormat("Criamos a pasta %s",working_folder+"\\"+folder_path);
      //--- redefinimos o código de erro
      ResetLastError();
      //--- execução bem-sucedida
      return true;
     }
   else
      PrintFormat("Falha ao cria a pasta %s. Código de erro %d",working_folder+folder_
```

© 2000-2024, MetaQuotes Ltd.

```
//--- execução mal-sucedida
   return false;
  }
```

Veja Também

FileOpen(), FolderClean(), FileCopy()

© 2000-2024, MetaQuotes Ltd.

FolderDelete

A função remove o diretório especificado. Se a pasta não estiver vazia, então ele não pode ser removida.

```
bool  FolderDelete(
   string  folder_name,        // String com o nome da pasta para deletar
   int     common_flag=0       // escopo
   );
```

Parâmetros

folder_name

[in] O nome do diretório que você deseja excluir. Contém o caminho completo para a pasta.

common_flag=0

[in] Bandeira determinando a localização do diretório. Se common_flag=FILE_COMMON, então o diretório está na pasta compartilhada para todos os terminais do cliente \Terminal\Common\Files. Caso contrário, o diretório está em uma pasta local (MQL5\Files ou MQL5\Tester\Files no caso de testes).

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Se o diretório contiver pelo menos um arquivo e / ou subdiretório, então este diretório não pode ser excluído, ele deve ser limpo primeiro. FolderClean() é usado para limpar a pasta de todos os seus arquivos ou sub-pastas.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_FolderDelete.mq5 |
//|                       Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Descrição
#property description "O roteiro mostra um exemplo de uso FolderDelete()."
#property description "Primeiro duas pastas são criados, um deles está vazia, a segun
#property description "Quando se tenta excluir uma pasta não-vazia, será retornado um

//--- Mostrar a caixa de diálogo de parâmetros de entrada quando se inicia o script
#property script_show_inputs
//--- Parâmetros de entrada
```

```
input string   firstFolder="empty";     // Uma pasta vazia
input string   secondFolder="nonempty";// A pasta, em que um arquivo será criado
string filename="delete_me.txt";        // O nome do arquivo que será criado na pasta s
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Escreva o manipulador de arquivo aqui
   int handle;
//--- Descubra em qual pasta estamos trabalhando
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- Uma mensagem de depuração
   PrintFormat("working_folder=%s",working_folder);
//--- Tentando criar uma pasta vazia em relação ao caminho MQL5\Files
   if(FolderCreate(firstFolder,0)) // 0 (zero) significa que estamos trabalhando na pa
     {
      //--- Insira o caminho completo para a pasta criada
      PrintFormat("Pasta %s foi criada",working_folder+"\\"+firstFolder);
      //--- Redefine o código de erro
      ResetLastError();
     }
   else
      PrintFormat("Falha ao criar a pasta %s. Código de erro %d",working_folder+"\\"+f

//--- Agora, crie uma pasta não-vazia utilizando a função FileOpen()
   string filepath=secondFolder+"\\"+filename;  // Formar caminho do arquivo que desej
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // Flag FILE_WRITE neste caso é obri
   if(handle!=INVALID_HANDLE)
      PrintFormat("Arquivo %s foi aberto para leitura",working_folder+"\\"+filepath);
   else
      PrintFormat("Falha para criar arquivo %s na pasta %s. Código de erro=",filename,

   Comment(StringFormat("Preparar para deletar pastas %s e %s", firstFolder, secondFol
//--- Uma pequena pausa de 5 segundos para ler uma mensagem no gráfico
   Sleep(5000); // Sleep() não pode ser utilizado em indicadores!

//--- Mostrar uma caixa de diálogo e pedir ao usuário
   int choice=MessageBox(StringFormat("Você quer apagar pastas %s e %s?", firstFolder,
                         "Deletando pastas",
                         MB_YESNO|MB_ICONQUESTION); //  Dois botões - "Yes" e "No"

//--- Executar uma ação dependendo da variante selecionada
   if(choice==IDYES)
     {
      //--- Excluir no gráfico o formulário de comentário
      Comment("");
      //--- Adicione uma mensagem dentro do jornal "Experts"
      PrintFormat("Tentando deletar pastas %s e %s",firstFolder, secondFolder);
      ResetLastError();
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Deletar uma pasta vazia
      if(FolderDelete(firstFolder))
         //--- A seguinte mensagem deve aparecer uma vez que a pasta está vazia
         PrintFormat("Pasta %s foi deletada com sucesso",firstFolder);
      else
         PrintFormat("Falha para deletar pasta %s. Código de erro=%d", firstFolder, Ge

      ResetLastError();
      //--- Deletar a pasta que contém um arquivo
      if(FolderDelete(secondFolder))
         PrintFormat("Pasta %s foi deletada com sucesso", secondFolder);
      else
         //--- A seguinte mensagem deve aparecer uma vez que a pasta contém um arquivo
         PrintFormat("Falha para deletar pasta %s. Código de erro=%d", secondFolder, G
     }
   else
      Print("Exclusão cancelada");
//---
  }
```

Também Veja

FileOpen(), FolderClean(), FileMove()

© 2000-2024, MetaQuotes Ltd.

FolderClean

A função exclui todos os arquivos em uma pasta especificada.

```
bool  FolderClean(
   string  folder_name,        // String com o nome da pasta eliminada
   int     common_flag=0       // escopo
   );
```

Parâmetros

folder_name

[in] O nome do diretório onde você deseja excluir todos os arquivos. Contém o caminho completo para a pasta.

common_flag=0

[in] Bandeira determinando a localização do diretório. Se common_flag = FILE_COMMON, então o diretório está na pasta compartilhada para todos os terminais de clientes \Terminal\Common\Files. Caso contrário, o diretório está numa pasta local (MQL5\Files ou MQL5\Tester\Files no caso dos testes).

Valor do Retorno

Retorna true se bem sucedido, caso contrário false.

Observação

Por razões de segurança, o trabalho com arquivos é estritamente controlada na língua MQL5. Arquivos com que as operações de arquivo são conduzidas usando os meios MQL5, não podem estar fora da caixa de proteção do arquivo.

Esta função deve ser usada com cuidado, uma vez que todos os arquivos e todos os subdiretórios são excluídos irremediavelmente.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderClean.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Descrição
#property description "O script mostra um exemplo de uso FolderClean()."
#property description "Primeiro, os arquivos são criados na pasta especificada utiliza
#property description "Então, antes que os arquivos sejam apagados, um aviso é mostrad

//--- Mostrar caixa de diálogo de parâmetros de entrada quando se inicia o script
#property script_show_inputs
//--- Parâmetros de entrada
input string   foldername="demo_folder";  // Criar uma pasta em MQL5/Files/
```

© 2000-2024, MetaQuotes Ltd.

```
input int      files=5;                  // O número de arquivos para criar e exclui
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
   string name="testfile";
//---Primeiro abrir ou criar arquivos na pasta de dados do terminal
   for(int N=0;N<files;N++)
     {
      //--- O nome do arquivo no formato demo_pasta\testfileN.txt'
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- Abrir um arquivo com a bandeira para a escrita, neste caso, o 'demo_pasta
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- Descubra se a função FileOpen() foi bem sucedida
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("Falha para criar arquivo %s. Código de erro",filemane,GetLastEr
         ResetLastError();
        }
      else
        {
         PrintFormat("Arquivo %s foi aberto com sucesso",filemane);
         //--- O arquivo aberto não é mais necessário, então fechá-lo
         FileClose(handle);
        }
     }

//--- Verifique o número de arquivos na pasta
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("Totalmente a pasta %s contém %d arquivos",foldername,k);

//--- Mostrar uma janela para perguntar ao usuário
   int choice=MessageBox(StringFormat("Você vai apagar %d arquivos da pasta %s. Você c
                         "Excluindo arquivos da pasta",
                         MB_YESNO|MB_ICONQUESTION); // Dois botões - "Sim" e "Não"
   ResetLastError();

//--- Executar uma ação dependendo da variante selecionada
   if(choice==IDYES)
     {
      //--- Começar deletar arquivos
      PrintFormat("Tentando deletar todos os arquivos da pasta %s",foldername);
      if(FolderClean(foldername,0))
         PrintFormat("Arquivos foram deletados com sucesso, %d arquivos que restaram r
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("Falha ao deletar arquivos da pasta %s. Código de erro %d",folder
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      PrintFormat("Exclusão cancelada");
//---
  }
//+------------------------------------------------------------------+
//| Retornar o número de arquivos na pasta especificada              |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- Se pelo menos um arquivo encontrado, procurar por mais arquivos
   if(handle!=INVALID_HANDLE)
     {
      //--- Mostrar o nome do arquivo
      PrintFormat("Arquivo %s encontrado",filename);
      //--- Aumentar o contador de arquivos/pastas encontrados
      count++;
      //--- Iniciar a pesquisa em todos arquivos/pastas
      while(FileFindNext(handle,filename))
        {
         PrintFormat("Arquivo %s encontrado",filename);
         count++;
        }
      //--- Não se esqueça de fechar o manipulador de pesquisa após a conclusão
      FileFindClose(handle);
     }
   else // Falha ao obter o manipulador
     {
      PrintFormat("Pesquisa nos arquivos na pasta %s falhou",path);
     }
//--- Retorna o resultado
   return count;
  }
```

Também Veja

FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Indicadores Customizados

Este é o grupo de funções utilizadas na criação dos indicadores personalizados. Estas funções não podem ser usadas para escrever Expert Advisors e Scripts.

Função	Ação
SetIndexBuffer	Vincula-se o buffer de indicador especificado com dinâmica unidimensional array do tipo double
IndicatorSetDouble	Define o valor da propriedade de um indicador do tipo double
IndicatorSetInteger	Define o valor da propriedade de um indicador do tipo int
IndicatorSetString	Define o valor da propriedade de um indicador do tipo string
PlotIndexSetDouble	Define o valor da propriedade de linha de um indicador do tipo double
PlotIndexSetInteger	Define o valor da propriedade de linha de um indicador do tipo int
PlotIndexSetString	Define o valor da propriedade de linha de um indicador do tipo string
PlotIndexGetInteger	Retorna o valor da propriedade de linha de um indicador do tipo inteiro

Propriedades de indicador podem ser definidas utilizando as diretivas de compilador ou o uso de funções. Para entender melhor isso, é recomendável que você estude estilos de indicador como exemplos.

Todos os cálculos necessários de um indicador personalizado devem ser colocados na função predeterminada OnCalculate(). Se você usar uma forma abreviada da função chamada OnCalculate(), como

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

então a variável *rates_total* contém o valor do número total de elementos da ordem de price[], passados como um parâmetro de entrada para calcular os valores do indicador.

Parâmetro *prev_calculated* é o resultado da execução de OnCalculate() na chamada anterior, que permite organizar um algoritmo salvo para calcular os valores do indicador. Por exemplo, se o valor corrente de rates_total = 1000, prev_calculated = 999, então talvez seja bastante para fazer os cálculos apenas para um valor de cada buffer indicador.

Se a informação sobre o tamanho da entrada da ordem de preço fosse disponibilizado, então isso levaria à necessidade de fazer cálculos para 1000 valores de cada buffer de indicador. Na primeira chamada do OnCalculate(), o valor prev_calculated = 0. Se a ordem de price[] foi alterado de alguma forma, então neste caso prev_calculated também é igual a 0 (zero).

O parâmetro *begin* mostra o número de valores iniciais da ordem de preço, que não contêm dados para o cálculo. Por exemplo, se o valor de Accelerator Oscillator (para os quais os primeiros 37 valores não são calculados) foram utilizados como um parâmetro de entrada, então begin = 37. Por exemplo, vamos considerar um simples indicador:

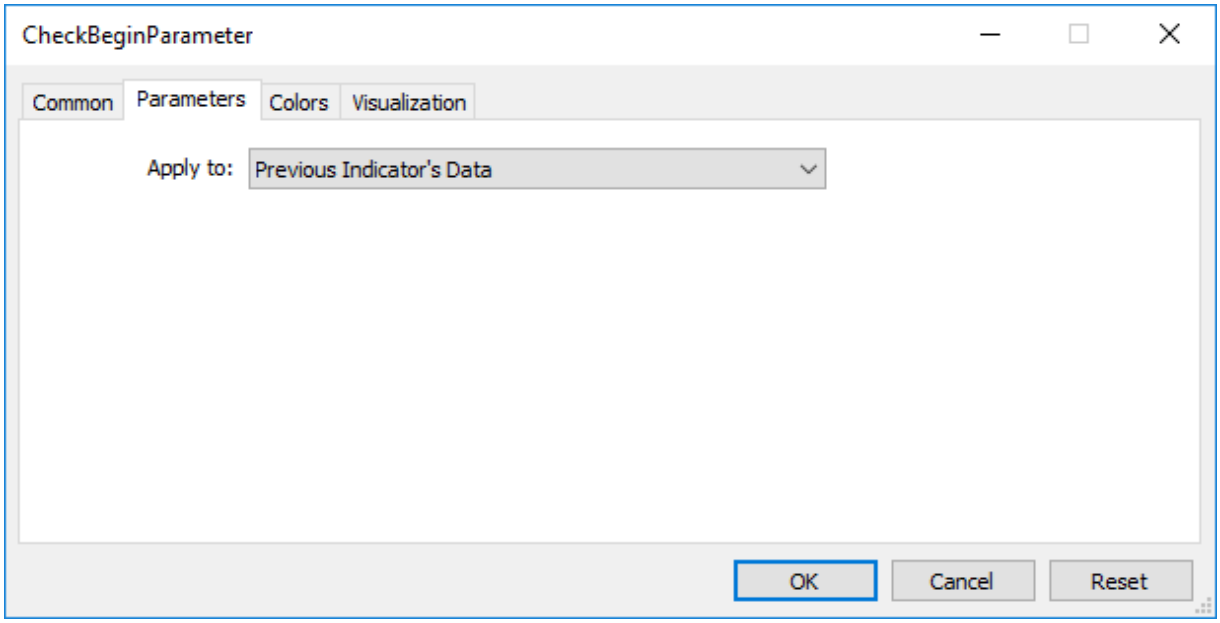
```
#property indicator_chart_window
```

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Etiqueta 1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers do indicador
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Arrastá-lo a partir da janela do "Navegador" para a janela do indicador Accelerator Oscillator e nós indicamos que os cálculos devem ser feitos com base nos valores do indicador anterior:

© 2000-2024, MetaQuotes Ltd.



Como um resultado, a primeira chamada do valor OnCalculate() de prev_calculated será igual a zero, e em outras chamadas será igual ao valor rates_total (até que o número de barras no gráfico de preço aumenta).

O valor do parâmetro begin será exatamente igual ao número das barras iniciais, para os quais os valores do indicador Accelerator não são calculados de acordo com a lógica do indicador. Se nós olharmos para o código fonte do indicador personalizado Accelerator.mq5, nós veremos as linha seguintes na função OnInit():

© 2000-2024, MetaQuotes Ltd.

```
//--- define a primeira barra a partir do qual o índice será desenhado
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

Usando a função PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values), nós definimos o número não-existente dos primeiros valores no array zero de um indicador personalizado, o qual não precisamos aceitar para o cálculo (empty_first_values). Assim, temos mecanismos para:

1. definir o número de valores iniciais de um indicador, o qual não deverá ser usado para os cálculos de outro indicador personalizado;
2. obter informações sobre o número dos primeiros valores a serem ignorados quando você chamar um outro indicador personalizado , sem entrar na lógica de seus cálculos.

© 2000-2024, MetaQuotes Ltd.

Estilos de Indicador em Exemplos

O MetaTrader 5 Terminal Cliente inclui 38 indicadores técnicos que podem ser usados em MQL5 programas usando funções apropriadas. Mas a principal vantagem da língua MQL5 é a capacidade de criar indicadores personalizados, os quais podem então serem utilizados em Expert Advisors ou simplesmente serem aplicados sobre gráficos de preços para fins de análise técnica.

O conjunto total dos indicadores podem serem derivados a partir de várias bases de estilos de desenhos, conhecidas como plotagem. Plotagem denota uma forma de visualização de dados, que o indicador calcula, armazena e providencia o pedido. Existem sete tipos básicos plotagem:

1. Uma linha
2. Uma seção (segmento)
3. Histograma
4. Seta (símbolo)
5. Uma área pintada (canal preenchido)
6. Bars
7. Castiçal (candlesticks) japonês

Cada plotagem requer um para cinco arrays do tipo double, em que os valores dos indicadores são armazenados. Para efeitos de comodidade, estes arrays são associados com os buffers do indicador. O número de buffers em um indicador deve ser declarado com antecedência usando diretivas de compilador, por exemplo:

```
#property indicator_buffers 3 // Número de buffers
#property indicator_plots   2 // Número de plotes
```

O número de buffers no indicador é sempre maior ou igual ao número de plotagem no indicador.

Como cada tipo de plotagem básica pode ter variação de cor ou detalhes de construção, o número real de tipos de plotagem em MQL5 é de 18:

Plotagem	Descrição	Buffers de valor	Buffers de cor
DRAW_NONE	Não é apresentado visualmente no gráfico, mas os valores do buffer correspondente podem ser vistos na Janela de Dados	1	-

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
DRAW_LINE	Uma linha é plotada sobre os valores do buffer correspondente (valores vazios no buffer são indesejáveis)	1	-
DRAW_SECTION	É desenhado como segmentos de linha entre os valores do buffer correspondente (geralmente tem muitos valores vazios)	1	-
DRAW_HISTOGRAM	É desenhado como um histograma a partir da linha zero para os valores do buffer correspondente (pode ter valores vazios)	1	-

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
DRAW_HISTOGRAM2	É desenhado como um histograma sobre dois buffers de indicador (pode ter valores vazios)	2	-
DRAW_ARROW	É desenhado como símbolos (pode ter valores vazios)	1	-
DRAW_ZIGZAG	Similar ao estilo DRAW_SECTION, mas diferentemente, pode plotar segmentos verticais em uma barra	2	-
DRAW_FILLING	Preenchimento de cor entre duas linhas. 2 valores dos buffers correspondentes são mostrado	2	-

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
	s na Janela de Dados		
DRAW_BARS	É desenhado como barras. 2 valores dos buffers correspondentes são mostrados na Janela de Dados	4	-
DRAW_CANDLES	Desenhado como Castiçais Japoneses (candlesticks). 2 valores dos buffers correspondentes são mostrados na Janela de Dados	4	-
DRAW_COLOR_LINE	Uma linha para o qual você pode alternar as cores em diferentes barras ou mudar suas	1	1

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
	cores, a qualquer momento		
DRAW_COLOR_SECTION	Similar ao estilo DRAW_SECTION, mas a cor de cada seção pode ser ajustada individualmente; cor também pode ser ajustada dinamicamente	1	1
DRAW_COLOR_HISTOGRAM	Similar ao estilo DRAW_HISTOGRAM, mas cada faixa pode ter uma cor diferente, você pode definir a cor dinamicamente	1	1
DRAW_COLOR_HISTOGRAM2	Similar ao estilo DRAW_HISTOGRAM2, mas cada faixa pode ter uma cor	2	1

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
	diferente, você pode definir a cor dinamicamente		
DRAW_COLOR_ARROW	Similar ao estilo DRAW_ARROW, mas cada símbolo pode ter sua cor. Cor pode ser mudada dinamicamente	1	1
DRAW_COLOR_ZIGZAG	O estilo DRAW_ZIGZAG com opções de cores individuais das seções e mudança dinâmica das cores	2	1
DRAW_COLOR_BARS	O estilo DRAW_BARS com opções de cores individuais das barras e mudança dinâmica das cores	4	1
DRAW_COLOR_CANDLES	O estilo DRAW_CANDLES	4	1

© 2000-2024, MetaQuotes Ltd.

Plotagem	Descrição	Buffers de valor	Buffers de cor
	com opções de cores individuais dos castiçais (candlesticks) e mudança dinâmica das cores		

A diferença entre um buffer de indicador e um array

Cada indicador, no seu nível global, você deve declarar uma ou mais arrays do tipo double, o qual então deve ser usado como um buffer de indicador usando a função SetIndexBuffer(). Para desenhar plotagem de indicador, apenas os valores dos buffers do indicador são utilizados, quaisquer outros arrays não podem ser usados para esta finalidade. Além disso, os valores de buffer são exibidos na Janela de Dados.

Um buffer de indicador deve ser dinâmico e não requer especificação de tamanho - o tamanho do array usado como buffer de indicador, é definido pelo subsistema do terminal com execução automática.

Após o array estar ligado ao buffer de indicador, a direção de indexação é definida por padrão num arrays comum, mas pode usar a função ArraySetAsSeries() para alterar o modo de acesso aos elementos array. Por padrão, o buffer de indicador é usado para armazenar dados usados para plotagem INDICATOR_DATA).

Se os cálculos dos valores de indicador requerem segurança de cálculos intermediários e armazenamento de valores adicionais para cada barra, então esse array pode ser declarado como um buffer de cálculo durante a ligação (INDICATOR_CALCULATIONS). Para valores intermediários, você também pode usar um array regular, mas neste caso, o programador tem que gerenciar o tamanho do array.

Algumas plotagens permitem definir uma cor para cada barra. Para armazenar a informação sobre cor, buffers de cores são usados (INDICATOR_COLOR_INDEX). A cor é um tipo integer color, mas todos os buffers de indicadores serão do tipo double. Valores de cor e buffers auxiliares (INDICATOR_CALCULATIONS) não podem ser obtidos usando CopyBuffer().

O número de buffers de indicadores devem ser especificados usando a diretiva de compilador #property indicator_buffers number_of_buffers:

```
#property indicator_buffers 3  //  o indicador tem 3 buffers
```

O número máximo permitido de buffers em um indicador é 512.

© 2000-2024, MetaQuotes Ltd.

Relevância de Buffers de Indicadores e Plotagem

Cada plotagem é baseada em um ou mais buffers do indicador. Assim, para a exibição de castiçais (candlesticks) simples, quatro valores são obrigatórios - Open (abertura), High (alta), Low (baixa) e Close (fechamento) de preços. Conseqüentemente, para exibir um indicador na forma de castiçais (candlesticks), é necessário declarar 4 buffers de indicadores e 4 arrays do tipo double para eles. Por exemplo:

```
//--- O indicador tem quatro buffers de indicadores
#property indicator_buffers 4
//--- O indicador tem uma plotagem
#property indicator_plots   1
//--- Graficamente plotando número 1 aparecerá como castiçais (candlesticks)
#property indicator_type1   DRAW_CANDLES
//--- Castiçais (candlesticks) serão desenhados em clrDodgerBlue
#property indicator_color1  clrDodgerBlue
//--- 4 arrays para buffers de indicadores
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Graficamente, plotagens automáticas usam buffers de indicadores de acordo com o número de plotagem. A numeração da plotagem começa com um, a numeração dos buffers começa com zero. Se a primeira plotagem requer 4 buffers do indicador, então os primeiros quatro buffers do indicador serão utilizados para desenhá-lo. Estes quatro buffers devem estar lincados com os arrays apropriados com a indexação correta usando a função SetIndexBuffer().

```
//--- Arrays vinculados com buffers de indicador
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // O primeiro buffer corresponde ao índi
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // O segundo buffer corresponde ao índic
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // O terceiro buffer corresponde ao índi
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // O quarto buffer corresponde ao índice
```

A plotagem dos castiçais (candlesticks), o indicador usará apenas o primeiro dos quatro buffers, porque a plotagem do "candlestick" foi anunciado sobre o primeiro número.

Alterar o exemplo, e adicionar plotagem de uma linha simples - DRAW_LINE. Agora, suponha que a linha é numerada como 1, e os castiçais (candlesticks) são o número 2. O número de buffers e o número plotagem aumentou.

```
//--- O indicador tem 5 buffers de indicador
#property indicator_buffers 5
//--- O indicador tem 2 plotagens
#property indicator_plots   2
//--- Plotagem 1 é uma linha
#property indicator_type1   DRAW_LINE
//--- A cor d linha é clrDodgerRed
#property indicator_color1  clrDodgerRed
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Plotagem 2 é desenhado como castiçal (candlestick) japonês
#property indicator_type2   DRAW_CANDLES
//--- A cor do castiçal (candlestick) é clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- 5 arrays para buffers de indicador
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

A ordem das plotagem mudou, e agora a linha vem em primeiro lugar, seguido dos castiçais (candlesticks) Japoneses. Portanto, a ordem dos buffers é apropriado - primeiro anunciamos um buffer para a linha com o índice zero, e então quatro buffers para os castiçais (candlesticks).

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // O primeiro buffer corresponde ao i
//--- Ligando arrays com buffers de indicador para os castiçais (candlesticks)
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // O segundo buffer corresponde ao ín
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // O terceiro buffer corresponde ao i
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // O quarto buffer corresponde ao índ
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // O quinto buffer corresponde ao índ
```

O número de buffers e plotagens serão definidos usando somente diretivas de compilador, é impossível alterar essas propriedades dinamicamente usando funções.

Versões Coloridas de Estilos

Como são vistos na tabela, os estilos são divididos em dois grupos. O primeiro grupo inclui estilos cujo nome não existe a palavra **COLOR**, nós chamamos de estilos básicos:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

No segundo grupo, os nomes de estilo contêm a palavra **COLOR**, vamos chamá-los de versões de cores:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM

© 2000-2024, MetaQuotes Ltd.

- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Todas as versões de cores de estilos diferentes a partir das básicas permitem especificar uma cor para cada parte da plotagem. A parte mínima de plotagem é uma barra, então podemos dizer que as versões de cores permitem definir a cor de cada bar.

As exceções são os estilos DRAW_NONE e DRAW_FILLING, eles não tem versões de cores.

Para definir a plotagem da cor em cada barra, um buffer adicional para armazenar o índice de cor foi adicionada à versão de cor. Estes índices indicam o número de uma cor numa array especial, a qual contém um conjunto predefinido de cores. O tamanho do array das cores é 64. Isto significa que cada versão de cor de um estilo de pintura permite uma plotagem de 64 cores diferentes.

O conjunto e o número de cores na array especial de cores será definido através de uma diretiva de compilação #property indicator_color, onde pode especificar todas as cores necessárias separadas por vírgulas. Por exemplo, como uma entrada de um indicador:

```
//--- Definir 8 cores para colorir os candlesticks (eles são armazenados na array espe
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

Ele afirma que para plotar 1, 8 cores são definidas, que serão colocados em um array especial. Além disso o programa não especifica a cor da plotagem, mas apenas o seu índice. Se quisermos definir a cor vermelha para o número de barras K, o valor da cor do índice a partir de array deve ser definido no buffer de cor do indicador. A cor vermelha é especificada pela primeira vez na diretiva, que corresponde ao número de índice 0.

```
  //--- definir a cor clrRed para candlestick
  col_buffer[buffer_index]=0;
```

O conjunto de cores não é colocado de uma vez por todas, pode ser alterado de forma dinâmica usando PlotIndexSetInteger(). Exemplo:

```
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          color_array[i]);      //  Uma nova cor
```

Propriedades do indicador e plotagem

Para plotagem de indicadores, propriedades podem ser definidas por meio de diretivas de compilador e usando as funções apropriadas. Ler maiores informações sobre isto em Conexão entre Propriedades do Indicador e Funções. Mudança dinâmica de propriedades do indicador usando funções especiais permitem a criação de indicadores personalizados mais flexíveis.

© 2000-2024, MetaQuotes Ltd.

Início do Desenho do Indicador no Gráfico

Em muitos casos, de acordo com as condições do algoritmo, é impossível iniciar o cálculo dos valores do indicador imediatamente com a barra atual, uma vez que é necessário proporcionar um número mínimo de barras anteriores disponíveis no histórico. Por exemplo, muitos tipos de alisamento implicam na utilização de um array de preços sobre N barras anteriores, e com base nestes valores, o valor do indicador na barra atual é calculado.

Nestes casos, não existe qualquer maneira para calcular os valores do indicador para as primeiras N barras, ou estes valores não se destinam a ser apresentados no gráfico e são apenas subsidiadas para calcular novos valores. Para evitar a plotagem do indicador nas primeiras N barras do histórico, defina o valor N para a propriedade PLOT_DRAW_BEGIN para a plotagem correspondente:

```
//--- Ligando arrays com buffers de indicador para os castiçais (candlesticks)
PlotIndexSetInteger(number_of_plot,PLOT_DRAW_BEGIN,N);
```

Onde:

- number_of_plot – um valor a partir de zero para indicator_plots-1 (numeração dos lotes inicia com zero).
- N - o número da primeira barra no histórico, no qual o indicador não deve ser exibido sobre o gráfico.

© 2000-2024, MetaQuotes Ltd.

DRAW_NONE

O estilo DRAW_NONE é projetado para uso em casos onde é necessário calcular os valores de um buffer e mostrá-los na Janela de Dados, mas a plotagem no gráfico não é necessária. Para configurar com precisão use a expressão IndicatorSetInteger(INDICATOR_DIGITS,num_chars) na função OnInit():

```
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Define a precisão dos valores a serem exibidos na Janela de Dados
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

O número de buffers requeridos para plotagem de DRAW_NONE é 1.

Um exemplo do indicador que mostra o número da barra na janela de dados, sobre a qual o mouse passa sobre ela. A numeração corresponde as séries temporais (timeseries), significando que a atual barra inacabada tem índice zero e a mais antiga barra tem o maior índice.

Note que apesar do fato de ser definida a plotagem para a cor vermelha **#1**, o indicador não desenha nada no gráfico.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar Invisível
#property indicator_label1  "Barra Índice"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- buffers do indicador
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Ligando um array e um buffer de indicador
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Define a precisão dos valores a serem exibidos na Janela de Dados
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- Se este for o primeiro cálculo do indicador
   if(prev_calculated==0)
     {
      //--- Renumerar as barras pela primeira vez
      CalcValues(rates_total,close);
      //--- Lembrar que a abertura do tempo da barra atual é no fechamento da última b
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
     }
```

```
   else
     {
      //--- Se uma nova barra apareceu, sua abertura de tempo difere da última barra
      if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
        {
         //--- Renumerar as barras mais uma vez
         CalcValues(rates_total,close);
         //--- Atualizar o tempo de abertura da barra atual com a última barra
         lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Número das barras como numa timeseries                           |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- Definir a indexação do buffer do indicador como numa timeseries
   ArraySetAsSeries(InvisibleBuffer,true);
//--- Definir cada barra com sua numeração
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

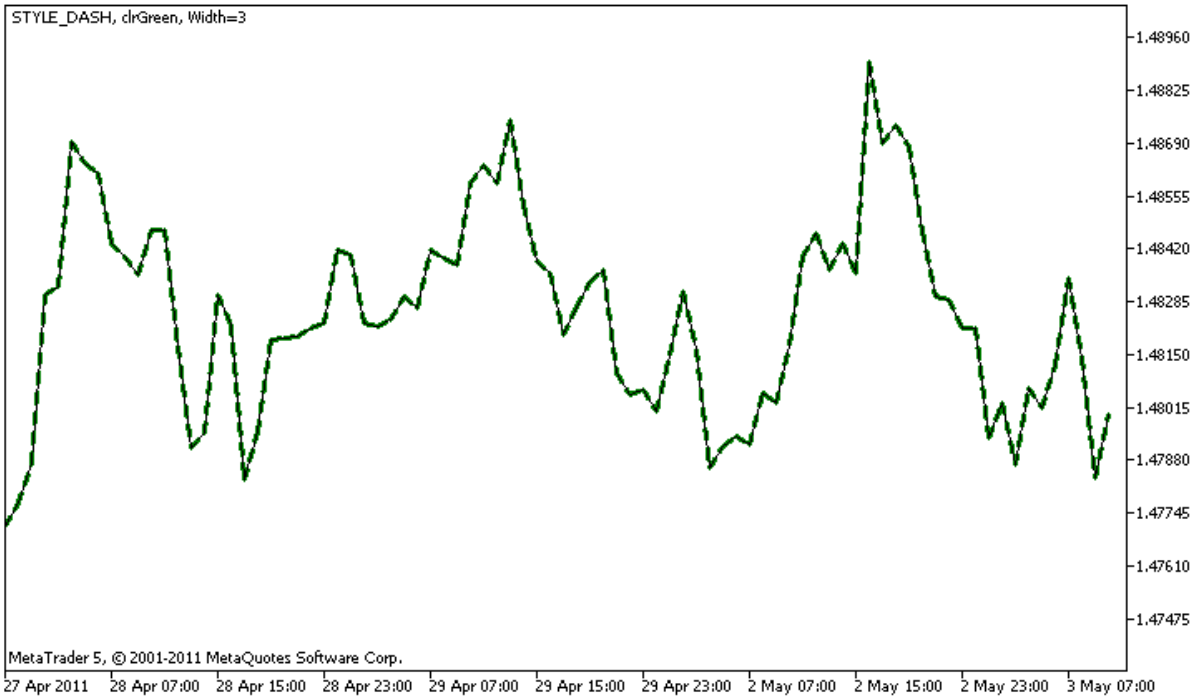
© 2000-2024, MetaQuotes Ltd.

DRAW_LINE

DRAW_LINE desenha uma linha de cor especifica pelos valores do buffer do indicador. A largura, estilo e cor da linha pode ser definido utilizando as diretivas de compilação e dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O número de buffers requerido para plotagem de DRAW_LINE é 1.

Um exemplo do indicador que desenha a linha usando o fechamento de preço das barras. A linha de cor, a largura e o estilo mudam aleatoriamente a cada N=5 ticks.



Note que inicialmente para **plot1** com DRAW_LINE, as propriedades são definidas usando a diretiva de compilador #property, e então na função OnCalculate() estas três propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_LINE"
#property description "Ele desenha uma linha de uma cor especifica no fechamento dos p
#property description "Cor, a largura e o estilo de linhas é alterado aleatoriamente"
#property description "depois de cada N ticks"
```

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Propriedades de linha são definidas usando as diretivas de compilação
#property indicator_label1  "Line"      // Nome de uma plotagem para a janela de dados
#property indicator_type1   DRAW_LINE   // Tipo de plotagem é uma linha
#property indicator_color1  clrRed      // Cor da linha
#property indicator_style1  STYLE_SOLID // Estilo da linha
#property indicator_width1  1           // Largura da linha
//--- parâmetros de entrada
input int      N=5;         // Número de ticks para mudar
//--- Um buffer do indicador para plotar
double         LineBuffer[];
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Ligando um array e um buffer de indicador
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- Inicializando o gerador de números pseudo-aleatórios
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Bloco para cálculo dos valores do indicador
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a aparência da linha desenhada no indicador               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a cor da linha
//--- Obtém aleatoriamente um número
   int number=MathRand();
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+(string)colors[color_index];

//--- Um bloco para alterar a largura da linha
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+", Width="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_SECTION

DRAW_SECTION desenha seções da cor específica pelos valores do buffer do indicador. A largura, cor e estilo da linha podem ser especificadas como para o estilo DRAW_LINE - usando as diretivas de compilação ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas aparências sejam alteradas dependendo da situação corrente.

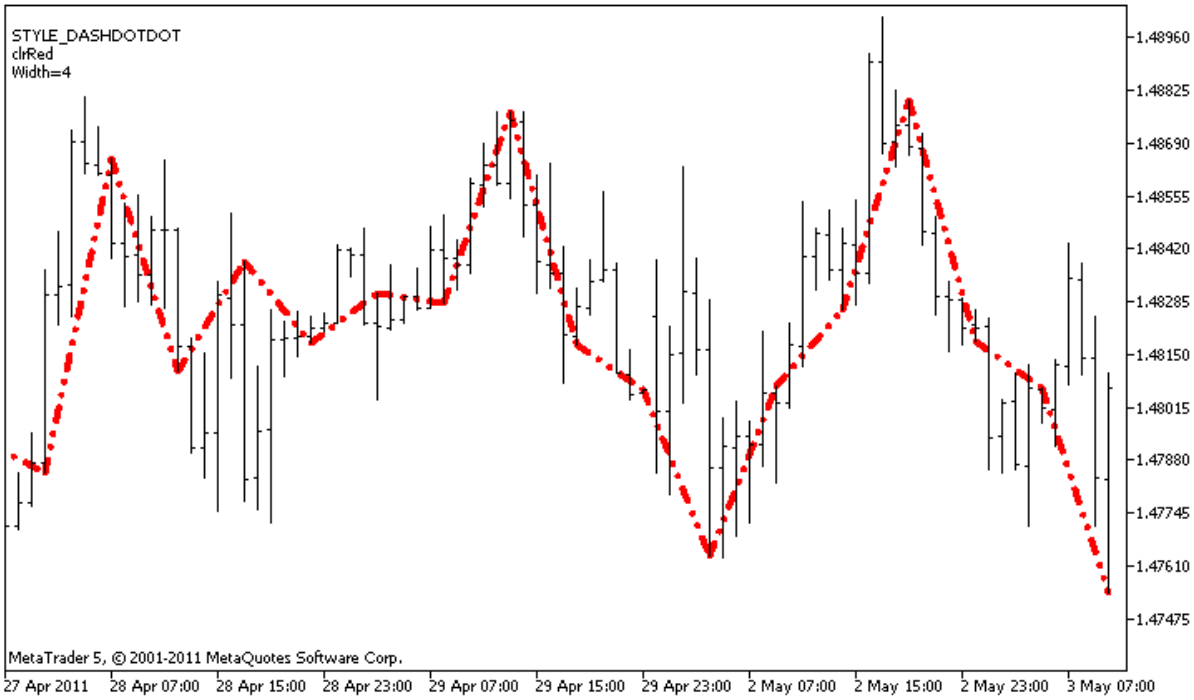
Seções são desenhadas a partir de um valor não vazio para outro valor não vazio do buffer de indicador, valores vazios são ignorados. Para especificar qual o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE: Por exemplo, se o indicador deve ser desenhado como uma seqüência de seções sobre valores diferentes de zero, então você precisa definir o valor zero como um vazio:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Sempre preencha explicitamente nos valores dos buffers do indicador, defina um valor vazio num buffer para os elementos que não devem ser plotados.

O número de buffers requeridos para plotagem de DRAW_SECTION é 1.

Um exemplo do indicador que desenha seções entre os preços de Alta (high) e de Baixa (low). A cor, a largura, e o estilo de **todas** seções mudam aleatoriamente a cada **N** ticks.



Note que para inicializar **plot1** com DRAW_SECTION, as propriedades são definidas usando a diretiva do compilador #property, e então na função OnCalculate(), estas três propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
```

```
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_SECTION"
#property description "Desenha seções em linha reta para cada barra"
#property description "A cor, a largura eo estilo de seções são alteradas aleatoriamen
#property description "depois de cada N ticks"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar Seção
#property indicator_label1  "Seção"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      bars=5;           // O comprimento das seções nas barras
input int      N=5;              // O número de ticks para mudar o estilo das seções
//--- Um buffer do indicador para plotar
double         SectionBuffer[];
//--- Uma variável auxiliar para calcular extremidades das seções
int            divider;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Ligando um array e um buffer de indicador
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Verifica o parâmetro do indicador
   if(bars<=0)
     {
      PrintFormat("Valor invalido do parâmetro da barra=%d",barras);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- O número de barra a partir da qual o cálculo dos valores do indicador inicia
   int start=0;
//--- Se o indicador foi calculado antes, então definir o início na barra anterior
   if(prev_calculated>0) start=prev_calculated-1;
//--- Aqui estão todos os cálculos dos valores do indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- Obter um resto da divisão do número de barra por 2*Barras
      int rest=i%divider;
      //--- Se o número de barra é divisível por 2*Barras
      if(rest==0)
        {
         //--- Definir o final da seção na alta de preço desta barra
         SectionBuffer[i]=high[i];
        }
      //---Se o resto da divisão é igual para barras,
      else
        {
         //--- Definir o final da seção na alta de preço desta barra
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- Se nada aconteceu, ignore a barra - definir 0
```

© 2000-2024, MetaQuotes Ltd.

```
          else SectionBuffer[i]=0;
         }
      }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Alterar a aparência das seções no indicador                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco de mudança da cor da linha
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para alterar a largura da linha
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a largura
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- define o estilo da linha
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

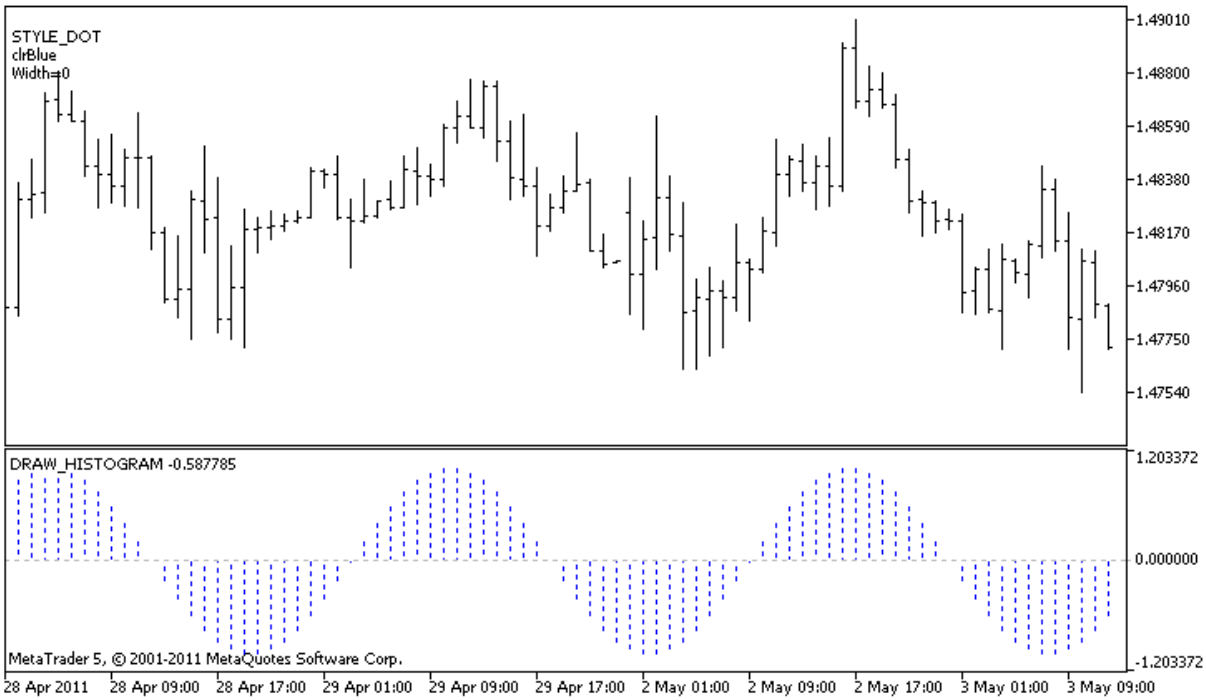
DRAW_HISTOGRAM

O estilo DRAW_HISTOGRAM desenha um histograma como uma seqüência de colunas de uma especifica cor do zero para um especifico valor. Os valores são extraídos do buffer do indicador. A largura, cor e estilo da coluna pode ser especificado como para o estilo DRAW_LINE - usando diretivas de compilação ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem alterar o visual do histograma com base na situação atual.

Uma vez que uma coluna a partir do nível zero é desenhada sobre cada barra, DRAW_HISTOGRAM deve ser melhor utilizada numa janela do gráfico em separado. Na maioria das vezes este tipo de de plotagem é usada para criar indicadores do tipo de oscilador, por exemplo, Bears Power ou OsMA. Para os valores vazios que não são visíveis, o valor zero deve ser especificado.

O número de buffers requerido para plotagem de DRAW_HISTOGRAM é 1.

Um exemplo do indicador que desenha uma senóide de uma cor específica com base na função MathSin(). A cor, a largura e o estilo de **todas** colunas do histograma mudam aleatoriamente a cada **N** ticks. O parâmetro de barras determina o período da senóide, isto é, após um número especifico de barras a senóide vai repetir o ciclo.



Note que inicialmente para **plot1** com DRAW_HISTOGRAM, as propriedades são definidas usando a diretiva de compilador #property, e então na função OnCalculate() estas três propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_HISTOGRAM"
#property description "Ele desenha um senóide como um histograma em uma janela separad
#property description "A cor e a largura das colunas são alterados aleatoriamente"
#property description "depois de cada N ticks"
#property description "O parâmetro das barras define o número de barras no ciclo da se

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar Histograma
#property indicator_label1  "Histograma"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      bars=30;          // O período de uma senóide em barras
input int      N=5;              // O número de ticks para alterar o histograma
//--- buffers do indicador
double         HistogramBuffer[];
//--- Um fator para obter o ângulo 2Pi em radianos, quando multiplicado pelo parâmetro
double    multiplier;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Calcular a multiplicação
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Defina o valor das barras=%d maior do que 1",bars);
      //--- Finalização antecipada do indicador
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Calcule os valores do indicador
   int start=0;
//--- Se já foi calculado durante os inícios anteriores do OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // definir o início do cálculo com a
//--- Preencher o buffer do indicador com valores
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a aparência das linhas no indicador                       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a cor da linha
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para alterar a largura da linha
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a largura
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- define o estilo da linha
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_HISTOGRAM2

O estilo DRAW_HISTOGRAM2 desenha uma histograma de uma cor especifica – segmentos verticais usando os valores de dois buffers do indicator. A largura, cor e estilo dos segmentos podem ser especificados para o estilo DRAW_LINE - usando diretivas de compilação ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem alterar o visual do histograma com base na situação atual.

O estilo DRAW_HISTOGRAM2 pode ser usado numa sub-janela separada e na sua janela principal. Para valores vazios nada é desenhado, todos os valores nos buffers do indicador precisam ser definidos explicitamente. Buffers não são inicializados com o valor zero.

O número de buffers requerido para plotagem de DRAW_HISTOGRAM2 é 2.

Um exemplo de indicador que plota um segmento vertical da cor especifica e largura entre a Abertura (open) e o Fechamento (close) de preços de cada barra. A cor, a largura e o estilo de **todas** as colunas do histograma mudam aleatoriamente a cada **N** ticks. durante a inicialização do indicador, na função OnInit(), o número do dia da semana em que não será desenhado o histograma - invisible_day - é definido aleatoriamente. Para este fim um valor vazio é definido PLOT_EMPTY_VALUE=0:

```
//--- Definir um valor vazio
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Note que inicialmente para **plot1** com DRAW_HISTOGRAM2, as propriedades são definidas usando a diretiva de compilador #property, e então na função OnCalculate() estas três propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                              DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Un indicador para demonstrar DRAW_HISTOGRAM2"
#property description "É desenhado um segmento entre a abertura e fechamento em cada k
#property description "A cor, a largura e o estilo são alterados aleatoriamente"
#property description "depois de cada N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;              // O número de ticks para alterar o histograma
//--- buffers do indicador
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- O dia da semana, em que o indicador não é plotado
int invisible_day;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- Definir um valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Obtém aleatoriamente um número a partir do 0 até 5
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Calcule os valores do indicador
   int start=0;
//--- Para obter o dia da semana pela abertura de preço de cada barra
   MqlDateTime dt;
//--- Se já foi calculado durante os inícios anteriores do OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // definir o início do cálculo com a
//--- Preencher o buffer do indicador com valores
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a aparência das linhas no indicador                       |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco de mudança da cor da linha
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para alterar a largura da linha
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Define a largura da linha
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- define o estilo da linha
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Adiciona informação sobre o dia que é omitido nos cálculos
   comm="\r\nDia não plotado - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_ARROW

O estilo DRAW_ARROW desenha setas de cor específica (símbolos do conjunto Wingdings) baseado no valor do buffer do indicador. A largura e a cor dos símbolos pode ser especificado semelhante ao modelo DRAW_LINE - usando diretivas de compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem alterar o visual de um indicador baseado na situação atual.

O código de símbolo é definido usando a propriedade PLOT_ARROW.

```
//--- Definir o código símbolo a partir das fontes Wingdings para desenhar em PLOT_ARR
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

O valor padrão do PLOT_ARROW=159 (um círculo).

Cada seta é na verdade um símbolo que tem a altura e o ponto de ancoragem e pode abranger algumas informações importantes num gráfico (por exemplo, o preço de fechamento da barra). Assim, podemos adicionalmente especificar o deslocamento vertical em pixels, o qual não depende da escala do gráfico. As setas serão deslocadas para baixo pelo número especificado de pixels, embora os valores do indicador permanecerão os mesmos:

```
//--- Definir o deslocamento vertical das setas em pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Um valor negativo de PLOT_ARROW_SHIFT significa a mudança de setas para cima, valores positivos deslocam a seta para baixo.

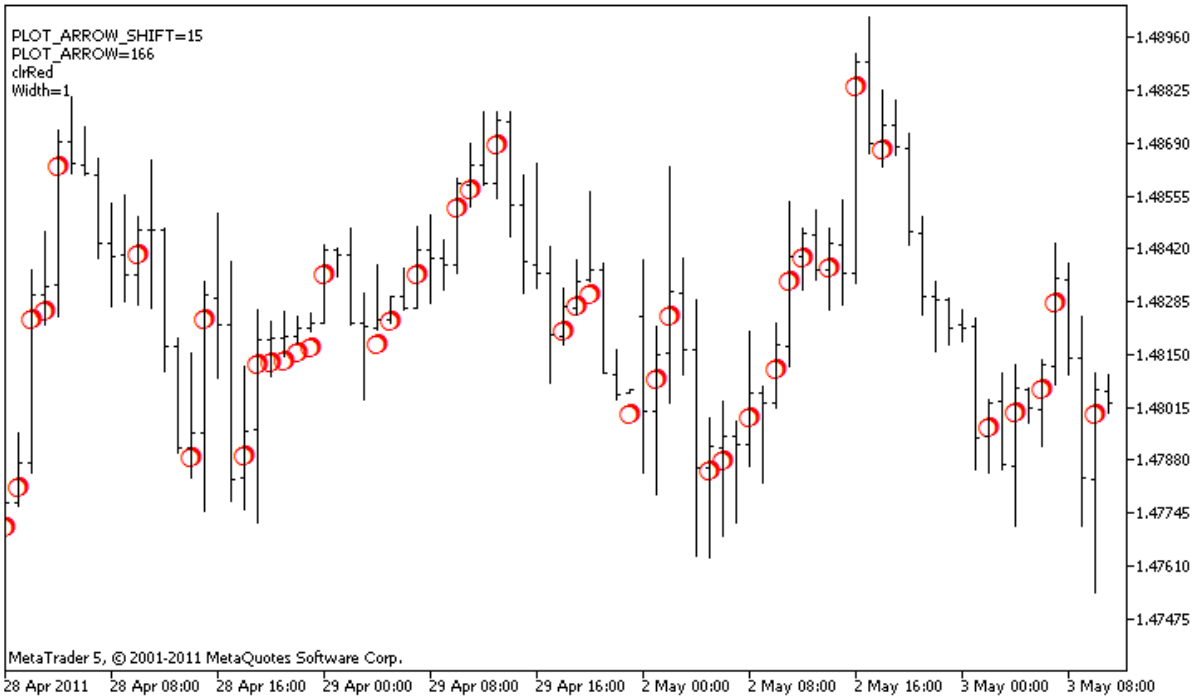
O estilo DRAW_ARROW pode ser usado numa sub-janela separada de um gráfico e em sua janela principal. Valores vazios não são desenhados e não aparecem na "Janela de Dados", todos os valores nos buffers do indicador devem ser definidos explicitamente. Buffers não são inicializados com o valor zero.

```
//--- Definir um valor vazio
   PlotIndexSetDouble(index_of_plot_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

O número de buffers necessários para plotagem DRAW_ARROW é 1.

Um exemplo de indicador que desenha as setas em cada uma das barras com o preço mais alto do que o preço final da barra anterior. A cor, a largura, deslocamento e símbolo de código de **todas** setas são alterados aleatoriamente a cada N ticks.

© 2000-2024, MetaQuotes Ltd.



No exemplo, para **plot1** com o estilo DRAW_ARROW, as propriedades, a cor e o tamanho são especificados usando a diretiva de compilador #property, e depois na função OnCalculate() as propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_ARROW"
#property description "Desenha setas definidas pelos caracteres Unicode, sobre um grá
#property description "A cor, o tamanho, deslocamento e código símbolo da seta, são a
#property description "depois de cada N ticks"
#property description "O parâmetro de código define o valor da base: código = 159 (um

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Plotagem de Setas
#property indicator_label1  "Setas"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- parâmetros de entrada
```

```
input int      N=5;         // Número de ticks para mudar
input ushort   code=159;    // Código símbolo para desenhar em DRAW_ARROW
//--- Um buffer do indicador para plotar
double         ArrowsBuffer[];
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- Definir o código símbolo para desenho em PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Definir o deslocamento vertical das setas em pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Definir como um valor vazio 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcular ticks para mudar a cor, tamanho, deslocamento e código da seta
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Bloco para cálculo dos valores do indicador
```

```
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- Calculando loop
   for(int i=1;i<rates_total;i++)
     {
      //--- Se o preço atual fechado é mais alto do que o anterior, desenhar uma seta
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- Otherwise specify the zero value
      else
         ArrowsBuffer[i]=0;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Alterar a aparência dos símbolos no indicador                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para formar as informações sobre as propriedades do indicador
   string comm="";
//--- Um bloco para alterar a cor da seta
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para mudar o tamanho das setas
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // O tamanho é definido a partir de 0 até 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a cor da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o código das setas (PLOT_ARROW)
   number=MathRand();
//--- Obter o restante da divisão de número inteiro para calcular um novo código da se
   int code_add=number%20;
//--- Definir o novo símbolo de código como o resultado de code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- Escreva o código de símbolo PLOT_ARROW
```

© 2000-2024, MetaQuotes Ltd.

```
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Um bloco de mudança do deslocamento vertical das setas em pixels
   number=MathRand();
//--- Obter o deslocamento como o restante da divisão inteira
   int shift=20-number%41;
//--- Definir o deslocamento novo de -20 a 20
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- Escrever o deslocamento PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_ZIGZAG

O estilo DRAW_ZIGZAG desenha segmentos da cor especifica baseada nos valor de dois buffers do indicador. Este estilo é muito similar ao DRAW_SECTION, mas ao contrário deste último, permite desenhar segmentos verticais dentro de uma barra, se os valores de ambos os buffers do indicador são definidos para este barra. Os segmentos são plotados a partir de um valor no primeiro buffer para um valor no segundo buffer do indicador. Nenhum dos buffers pode conter somente valores vazios, uma vez que neste caso nada é plotado.

A largura, cor e estilo da linha podem ser especificadas como para o estilo DRAW_SECTION - usando as diretivas de compilação ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas aparências sejam alteradas dependendo da situação corrente.

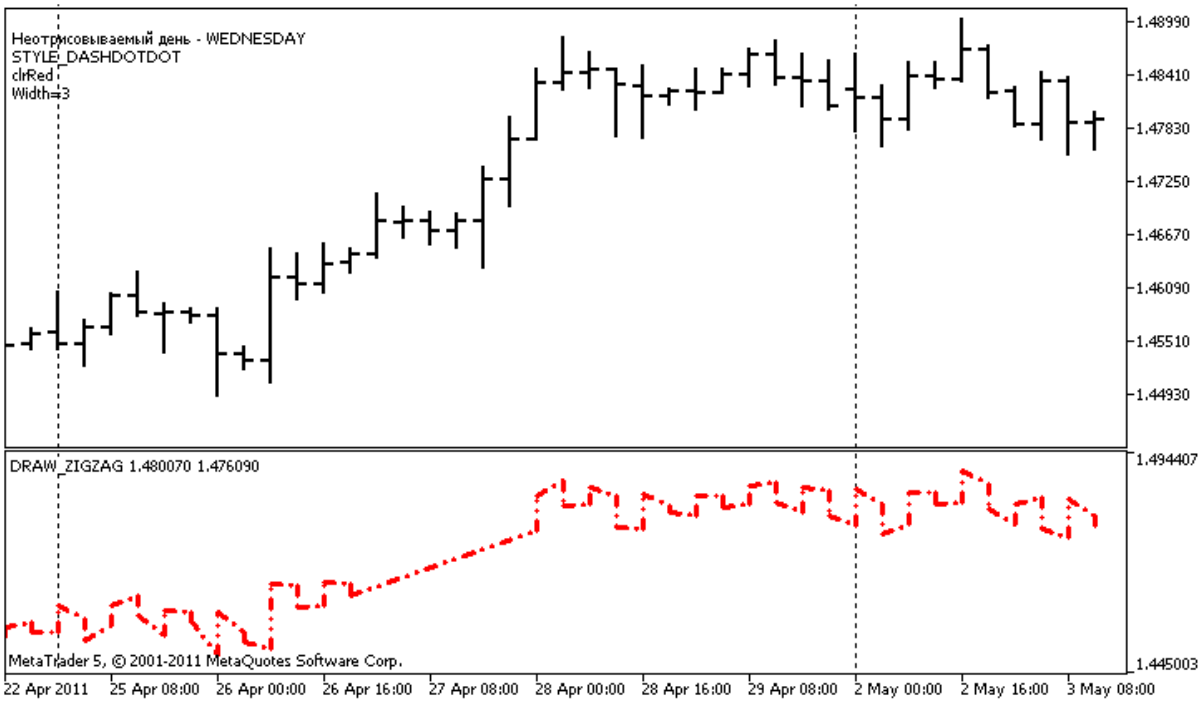
Seções são desenhadas a partir de um valor não vazio de um buffer para um valor não vazio de outro buffer do indicador. Para especificar o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Sempre preencher explicitamente os valores dos buffers do indicador, definir um valor vazio num buffer para ignorar barras.

O número de buffers requeridos para plotagem DRAW_ZIGZAG é 2.

Um exemplo de indicador de que plota uma serra com base nos altas e baixas dos preços. A cor, a largura eo estilo das linhas zigzag alteram aleatoriamente a cada **N** ticks.



Note que para inicializar **plot1** com DRAW_ZIGZAG, as propriedades são definidas usando a diretiva de compilar #property, e então na função OnCalculate() estas propriedades são definidas aleatoriamente.

© 2000-2024, MetaQuotes Ltd.

O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_ZIGZAG"
#property description "Ele desenha uma \"serra\" como segmentos de reta, ignorando as
#property description "O dia a ignorar é selecionado aleatoriamente durante a inicial
#property description "A cor, a largura e tamanho dos segmentos são alterados aleator
#property description " a cada N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar ZigZag
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;              // Número de ticks para alterar
//--- buffers do indicador
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- O dia da semana, em que o indicador não é plotado
int invisible_day;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen};
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Arrays de ligação e buffers do indicador
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- Obter um valor aleatório de 0 para 6, para este dia o indicador não é plotado
   invisible_day=MathRand()%6;
//--- O valor 0 (vazio) não vai participar no desenho
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- A estrutura de tempo é necessário para obter o dia da semana de cada barra
   MqlDateTime dt;

//--- A posição inicial dos cálculos
   int start=0;
//--- Se o indicador foi calculado com o tick anterior, então inicia o cálculo com o p
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Calculando loop
   for(int i=start;i<rates_total;i++)
     {
      //--- Escreva o tempo de abertura da barra na estrutura
      TimeToStruct(time[i],dt);
      //--- Se o dia da semana desta barra é igual para invisible_day
      if(dt.day_of_week==invisible_day)
        {
         //--- Escrever valores vazios de buffers para este barra
         ZigZagBuffer1[i]=0;
```

```
          ZigZagBuffer2[i]=0;
        }
      //--- Se o dia da semana é OK, preencha os buffers
      else
        {
         //--- Se o número de barra é o mesmo
         if(i%2==0)
           {
            //---  Escrever High (alta) no primeiro buffer a Low (baixa) no segundo
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- O número da barra é impar
         else
           {
            //--- Preenchimento da barra na ordem inversa
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Alterar a aparência dos segmentos zigzag                         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de ZigZag
   string comm="";
//--- Um bloco para mudar a cor do ziguezague
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para alterar a largura da linha
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
```

© 2000-2024, MetaQuotes Ltd.

```
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Adiciona informação sobre o dia que é omitido nos cálculos
   comm="\r\nDia não plotado - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_FILLING

O estilo DRAW_FILLING plota uma área colorida entre valores de dois buffers do indicador. De fato, este modelo desenha duas linhas e preenche o espaço entre elas, com uma das duas cores específicas. Ele é usado para a criação de indicadores que desenham canais. Nenhum dos buffers podem conter apenas valores vazios, uma vez que, neste caso, nada é plotado.

Você pode definir duas cores de preenchimento:

- a primeira cor é usada para as áreas onde os valores do primeiro buffer são maiores do que os valores do segundo buffer do indicador;
- a segunda cor é usada para as áreas onde os valores de do segundo buffer do são maiores do que os valores no primeiro buffer indicador.

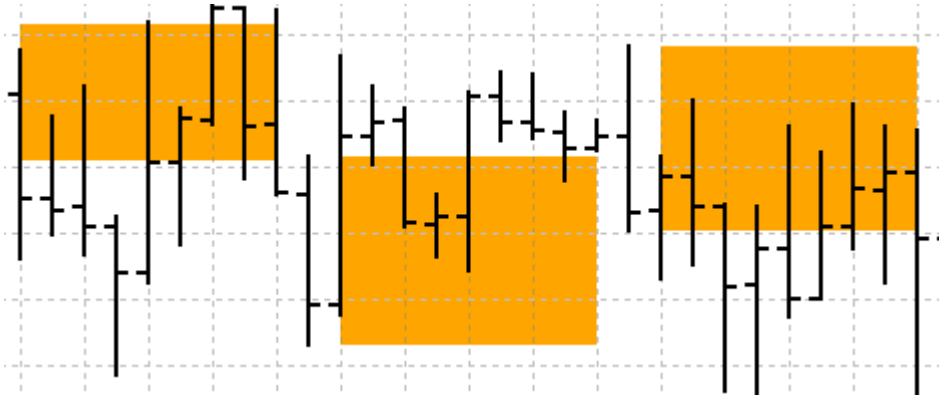
A cor de preenchimento pode ser definida usando as diretivas do compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O indicador é calculado para todas as barras, para a qual os valores dos dois buffers do indicador não são iguais a 0 ou o valor vazio. Para especificar o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- INDICATOR_EMPTY_VALUE (valor vazio) não vai participar no cálculo de
   PlotIndexSetDouble (DRAW_FILLING_creation_index,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VA
```

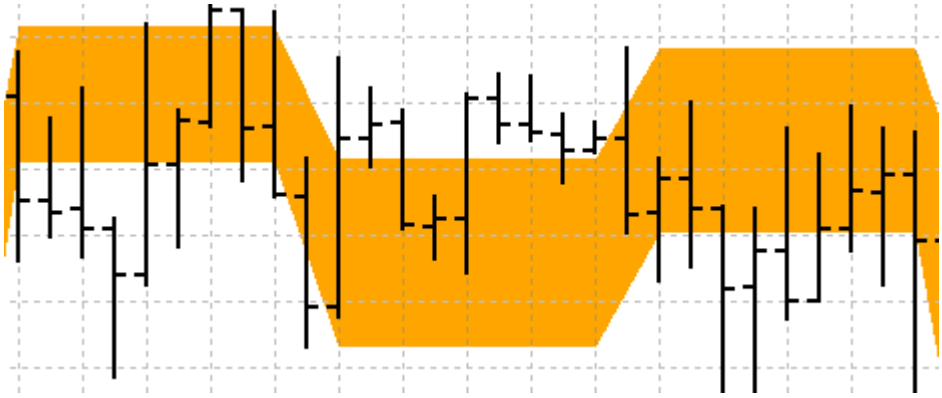
Desenhar sobre as barras que não participam no cálculo do indicador depende dos valores nos buffers do indicador:

- Barras, para os quais os valores de ambos os buffers do indicador são iguais a 0, não participam no desenho do indicador. Isto significa que a área com valores iguais a zero não é preenchida.



- Barras, para as quais os valores dos buffers do indicador são iguais ao "valor vazio", participam no desenho do indicador. A área com valores vazios será preenchida de modo que conecte as áreas com valores significativos.

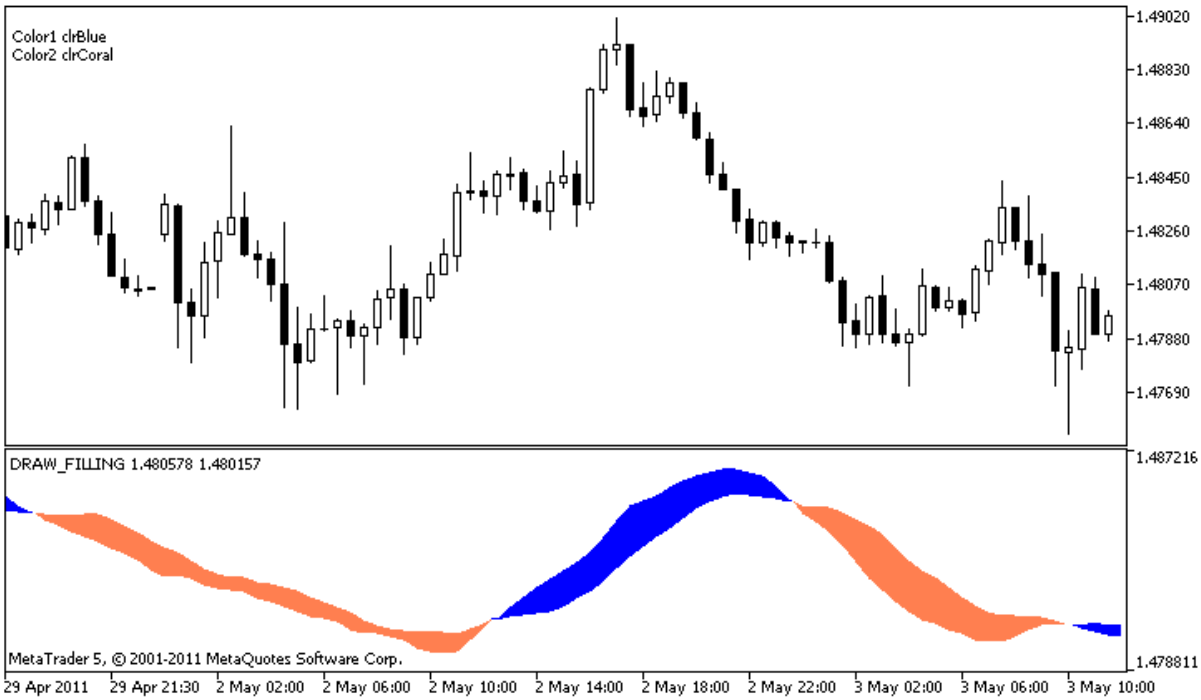
© 2000-2024, MetaQuotes Ltd.



Deve ser notado que se o "valor vazio" é igual a zero, as barras que não participam no cálculo do indicador também são preenchidas.

O número de buffers necessários para plotagem DRAW_FILLING é 2.

Um exemplo do indicador que desenha um canal entre duas MAs com diferentes períodos médios em uma janela separada. A mudança das cores no cruzamento das médias móveis mostra visualmente a mudança das tendências ascendentes e descendentes. As cores alteram aleatoriamente a cada **N** ticks. O parâmetro N é definido nos parâmetro externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).



Note que inicialmente para **plot1** com DRAW_FILLING as propriedades são definidas usando a diretiva de compilador #property, e então na função OnCalculate() novas cores são definidas aleatoriamente.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_FILLING"
#property description "Ele desenha um canal entre duas MAs em uma janela separada"
#property description "O preenchimento da cor é alterada aleatoriamente"
#property description "depois de cada N ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar Interseção
#property indicator_label1  "Interseção"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- parâmetros de entrada
input int      Fast=13;          // Período rápido da MA
input int      Slow=21;          // Período lento da MA
input int      shift=1;          // Deslocamento das MAs para frente (positivo)
input int      N=5;              // Número de ticks para alterar
//--- buffers do Indicador
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Faz o primeiro cálculo do indicador, ou os dados foram alterados e requerem um r
   if(prev_calculated==0)
     {
      //--- Copia todos os valores dos indicadores para os buffers apropriados
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // Preenche apenas os dados que são atualizados
     {
      //--- Obtém a diferença entre as barras atuais e as que iniciaram anteriormente
      int to_copy=rates_total-prev_calculated;
      //--- Se não existe qualquer diferença, ainda cópia um valor - na barra zero
      if(to_copy==0) to_copy=1;
      //--- copia valores to_copy para o fim dos buffers do indicador
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera as cores do preenchimento do canal                        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a cor da linha
```

```
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);

//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index1=number%size;
//--- Define a primeira cor como a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- Escreve a primeira cor
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   number=MathRand(); // Obtém um número aleatório
   int color_index2=number%size;
//--- Define a segunda cor como a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- Escreve a segunda cor
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_BARS

O estilo DRAW_BARS desenha barras nos valores de quatro buffers do indicador, que contêm a Abertura (open), Alta (high), Baixa (low) e Fechamento (close) dos preços. É usado para a criação de indicadores personalizados como barras, incluindo aqueles numa sub-janela separada de um gráfico e sobre outros instrumentos financeiros.

A cor das barras pode ser definida usando as diretivas de compilador ou dinamicamente utilizando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O indicador é desenhado apenas para as barras, para o qual os valores não vazios de **todos** quatro buffers do indicador são definidos. Para especificar o valor que deve ser considerado como "vazio", defina este valor na propriedade PLOT_EMPTY_VALUE:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Sempre preencher explicitamente os valores dos buffers do indicador, definir um valor vazio num buffer para ignorar barras.

O número de buffers necessários para plotagem de DRAW_BARS é 4. Todos os buffers para a plotagem devem ir um após o outro na ordem dada: Open (abertura do preço), High (alta do preço), Low (baixa do preço) e Close (fechamento do preço). Nenhum dos buffers podem conter apenas valores vazios, uma vez que, neste caso, nada é plotado.

Um exemplo do indicador que desenha barras sobre um instrumento financeiro selecionado numa janela separada. A cor das barras muda aleatoriamente a cada **N** ticks. O parâmetro N é definido nos parâmetro externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

© 2000-2024, MetaQuotes Ltd.

Por favor, note que para **plot1** com o estilo DRAW_BARS, a cor é definida usando a diretiva de compilador #property, e então, na função OnCalculate() a cor é definida aleatoriamente a partir de uma lista preparada anteriormente.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_BARS"
#property description "Ele desenha barras de uma seleção de símbolos numa janela sepa
#property description "A cor e a largura das barras, bem como o símbolo são alterados
#property description "cada N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plotar Barras
#property indicator_label1  "Barras"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;              // O número de ticks para alterar o tipo
input int      bars=500;         // O número de barras para mostrar
input bool     messages=false;   // Mostrar mensagens no log "Expert Advisors"
//--- buffers do Indicador
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- Nome do símbolo
string symbol;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Se as barras são muito pequenas - concluir o trabalho antes do tempo
   if(bars<50)
     {
      Comment("Por favor, especifique um número maior de barras! A operação do indica
```

```
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- O nome do símbolo, para o qual as barras são desenhadas
   symbol=_Symbol;
//--- Definir a exibição do símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- Um valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Selecione um novo símbolo a partir da janela de observação do Mercado
      symbol=GetRandomSymbolName();
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();

      int tries=0;
      //--- Faça 5 tentativas de preencher os buffers com os preços a partir símbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- Um contador de chamadas da função CopyFromSymbolToBuffers()
         tries++;
        }
```

```
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher os buffers do indicador com preços                     |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- No array de rates[], vamos copiar Open (abertura), High (alta), Low (baixa) e Cl
   MqlRates rates[];
//--- O contador de tentativas
   int attempts=0;
//--- Quanto foi copiado
   int copied=0;
//--- Fazer 25 tentativas para obter uma série temporal(timeseries) sobre o símbolo de
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s Copiar taxas(%s) tentativas=%d",__FUNCTION__,nome,t
     }
//--- Se falhou em copiar um número suficiente de barras
   if(copied!=bars)
     {
      //--- Formar uma mensagem string
      string comm=StringFormat("Para o símbolo %s, conseguiu receber somente %d barras
                               name,
                               copied,
                               bars
                               );
      //--- Mostrar uma mensagem num comentário na janela do gráfico principal
      Comment(comm);
      //--- Mostrar a mensagem
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Definir a exibição do símbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- Inicializar buffers com valores vazios
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayInitialize(BarsBuffer4,0.0);
//--- Copiar os preços para os buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcular o índice apropriado para os buffers
      int buffer_index=total-copied+i;
      //--- Escreva os preços para os buffers
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Aleatoriamente retorna um símbolo de Observação do Mercado   |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- O número de símbolos mostrada na janela de observação do Mercado
   int symbols=SymbolsTotal(true);
//--- A posição de um símbolo na lista - um número aleatório de 0 para símbolos
   int number=MathRand()%symbols;
//--- Retornar o nome de um símbolo na posição especificada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Altera a aparência das barras                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades da barra
   string comm="";
//--- Um bloco para alterar a cor das barras
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor da linha
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Um bloco para alterar a largura das barras
   number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Escreve o nome do símbolo
   comm="\r\n"+symbol+comm;

//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_CANDLES

O estilo DRAW_CANDLES desenha candlesticks sobre os valores dos quatro buffers do indicador, que contêm Open (abertura), High (alta), Low (baixa) e Close (fechamento) de preços. É usado para a criação de indicadores personalizados como uma seqüência de candlesticks, incluindo os inseridos numa sub-janela separada de um gráfico e noutros instrumentos financeiros.

A cor dos candlesticks pode ser definida usando as diretivas de compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O indicador é desenhado apenas para as barras, para o qual os valores não vazios de **todos** quatro buffers do indicador são definidos. Para especificar o valor que deve ser considerado como "vazio", defina este valor na propriedade PLOT_EMPTY_VALUE:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Sempre preencher explicitamente os valores dos buffers do indicador, definir um valor vazio num buffer para ignorar barras.

O número de buffers necessários para plotagem de DRAW_CANDLES é 4. Todos os buffers para a plotagem devem ir um após o outro na ordem dada: Open (abertura do preço), High (alta do preço), Low (baixa do preço) e Close (fechamento do preço). Nenhum dos buffers podem conter apenas valores vazios, uma vez que, neste caso, nada é plotado.

Para o estilo DRAW_CANDLES, é possível definir de uma a três cores, dependendo de isso é alterada a aparência das velas. Se for definida apenas uma cor, todas as velas no gráfico serão pintadas completamente com essa cor.

```
//--- velas semelhantes pintadas numa cor
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- como é indicada apenas uma cor, todas as velas terão uma cor
#property indicator_color1  clrGreen
```

Si nós especificarmos duas cores separadas por uma vírgula, os contornos das velas serão desenhados com a primeira cor e o corpo com a segunda.

```
//--- a cor das velas é diferente da cor das sombras
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- sombras e contornos das velas de cor verde e corpo branco
#property indicator_color1  clrGreen,clrWhite
```

Para exibir de maneira diferente as velas crescentes e decrescentes, é indispensável definir, separando por vírgulas, três cores. Neste caso, o contorno da vela será traçado usando a primeira cor, enquanto a cor da velas de alta e baixa serão definidas pela segunda e terceira cor.

```
//--- a cor das velas é diferente da cor das sombras
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- sombras e contorno de cor verde, corpo da vela de alta de cor branca, corpo da v
```

© 2000-2024, MetaQuotes Ltd.

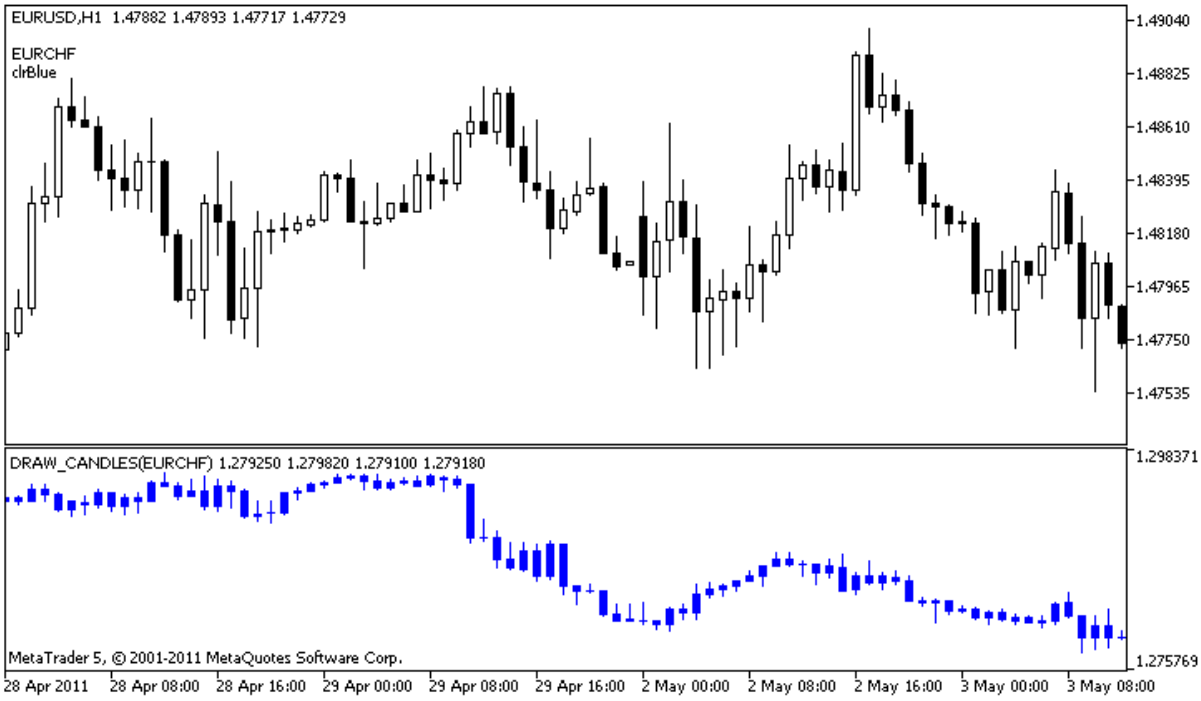
```
#property indicator_color1  clrGreen,clrWhite,clrRed
```

Assim, usando o estilo DRAW_CANDLES, é possível criar variantes personalizadas das velas. Também é possível alterar dinamicamente todas as cores -no processo de trabalho do indicador- mediante a função PlotIndexSetInteger(índice_de_construção_DRAW_CANDLES, PLOT_LINE_COLOR, número_de_modificador, cor), aqui o número_de_modificador pode ter os seguintes valores:

- 0 - cor do contorno e sombras
- 1- cor do corpo da vela de alta
- 2 - cor do corpo da vela de baixa

```
//--- definimos a cor do contorno e sombras
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- definimos a cor do corpo para velas de alta
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- definimos a cor do corpo para velas de baixa
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

Um exemplo de indicador que desenha candlesticks para um instrumento financeiro selecionado em uma janela separada. A cor dos candlesticks muda aleatoriamente a cada **N** ticks. O parâmetro N é definido nos parâmetro externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).



Por favor, note que para **plot1**, a cor é definida usando a diretiva de compilador #property, e então, na função OnCalculate() a cor é definida aleatoriamente a partir de uma lista preparada anteriormente.

```
//+------------------------------------------------------------------+
//|                                                DRAW_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
```

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_CANDLES."
#property description "ele desenha candlesticks de um símbolo selecionado numa janela
#property description " "
#property description "A cor e a largura dos candlesticks, bem como o símbolo são alte
#property description "aleatoriamente a cada N ticks"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plotar Barras
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parâmetros de entrada
input int      N=5;              // O número de ticks para alterar o tipo
input int      bars=500;         // O número de barras para mostrar
input bool     messages=false;   // Mostrar mensagens no log "Expert Advisors"
//--- buffers do Indicador
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- Nome do símbolo
string symbol;
//--- Um array para armazenar as cores
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Se as barras são muito pequenas - concluir o trabalho antes do tempo
   if(bars<50)
     {
      Comment("Por favor, especifique um número maior de barras! A operação do indicad
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
```

```
//--- Um valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- O nome do símbolo, para o qual as barras são desenhadas
   symbol=_Symbol;
//--- Definir a exibição do símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Selecione um novo símbolo a partir da janela de observação do Mercado
      symbol=GetRandomSymbolName();
      //--- Alterar a forma
      ChangeLineAppearance();
      //--- Selecione um novo símbolo a partir da janela de observação do Mercado
      int tries=0;
      //--- Faça 5 tentativas de preencher os buffers de Plot1 com os preços a partir
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- Um contador de chamadas da função CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Preenche o candlestick especificado                              |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
  {
//--- No array de rates[], vamos copiar Open (abertura), High (alta), Low (baixa) e Cl
   MqlRates rates[];
//--- O contador de tentativas
   int attempts=0;
//--- Quanto foi copiado
   int copied=0;
//--- Fazer 25 tentativas para obter uma série temporal(timeseries) sobre o símbolo de
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s Copiar taxas(%s) tentativas=%d",__FUNCTION__,nome,t
     }
//--- Se falhou em copiar um número suficiente de barras
   if(copied!=bars)
     {
      //--- Formar uma mensagem string
      string comm=StringFormat("Para o símbolo %s, conseguiu receber somente %d barras
                               name,
                               copied,
                               bars
                               );
      //--- Mostrar uma mensagem num comentário na janela do gráfico principal
      Comment(comm);
      //--- Mostrar a mensagem
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Definir a exibição do símbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- Inicializar buffers com valores vazios
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
```

© 2000-2024, MetaQuotes Ltd.

```
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- Sobre cada tick é copiado preços para os buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcular o índice apropriado para os buffers
      int buffer_index=total-copied+i;
      //--- Escreva os preços para os buffers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Aleatoriamente retorna um símbolo de Observação do Mercado   |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- O número de símbolos mostrada na janela de observação do Mercado
   int symbols=SymbolsTotal(true);
//--- A posição de um símbolo na lista - um número aleatório de 0 para símbolos
   int number=MathRand()%symbols;
//--- Retornar o nome de um símbolo na posição especificada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Altera a aparência das barras                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades da barra
   string comm="";
//--- Um bloco para alterar a cor das barras
   int number=MathRand(); // Obter um número aleatório
//--- O divisor é igual ao tamanho do array colors[]
   int size=ArraySize(colors);
//--- Obter o índice para selecionar uma nova cor como o restante da divisão inteira
   int color_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Escrever a cor
   comm=comm+"\r\n"+(string)colors[color_index];
//--- Escreve o nome do símbolo
   comm="\r\n"+symbol+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_LINE

O valor DRAW_COLOR_LINE é uma variante colorida do estilo DRAW_LINE; ele também desenha uma linha usando os valores do buffer do indicador. Mas esse estilo, como todos os estilos de cores com a palavra **COLOR** em seu título tem um buffer indicador adicional e especial que armazena o índice de cores (número) a partir de um conjunto especial array de cores. Assim, a cor de cada segmento de linha pode ser definida através da especificação do índice de cores para desenhar a linha naquela barra.

A largura, o estilo e as cores das linhas podem ser definidas usando as diretivas de compilador e dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

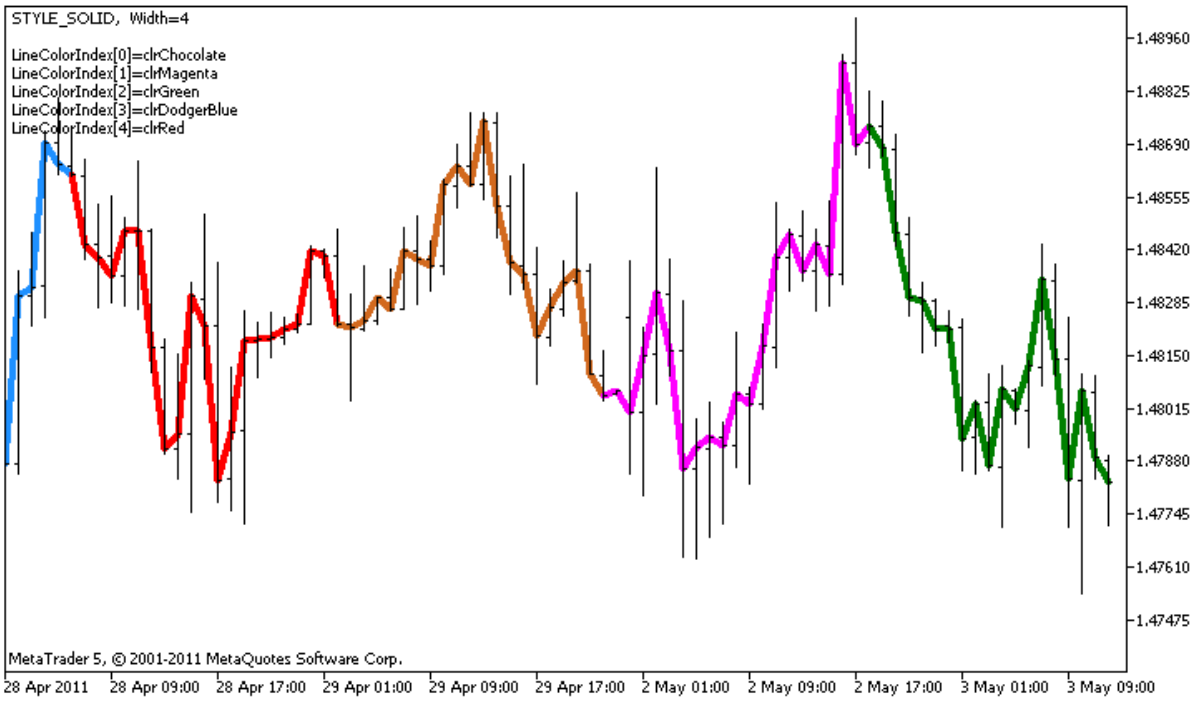
O número de buffers requerido para plotagem de DRAW_COLOR_LINE é 2.

- um buffer para armazenar os valores do indicador utilizado para desenhar uma linha;
- um buffer para armazenar o índice de cores da linha em cada barra.

Cores serão especificadas pela diretiva de compilador #property indicator_color1 separada por uma vírgula. O número de cores não pode exceder a 64.

```
//--- Define 5 cores para colorir cada barra (elas são armazenadas no array especial)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Podem se
```

Um exemplo do indicador que desenha a linha usando o fechamento de preço das barras. A largura da linha e estilo são alterados aleatoriamente a cada N=5 ticks.



As cores dos segmentos de linhas também são alteradas aleatoriamente na função personalizada ChangeColors().

```
//+------------------------------------------------------------------+
```

```
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

O exemplo mostra a característica das versões "cor" de indicadores - para mudar a cor de um segmento de linha, você não precisa alterar os valores no buffer ColorLineColors[] (que contém os índices de cores). Tudo que você precisa fazer é definir novas cores em um array especial. Isto permite-lhe alterar rapidamente a cor de uma vez para toda a plotagem, mudando apenas um pequeno array de cores utilizando a função PlotIndexSetInteger().

Note que para inicializar **plot1** com DRAW_COLOR_LINE as propriedades são definidas usando a diretiva de compilador #property, e então na função OnCalculate() existem três propriedades que são definidas aleatoriamente.

O N e Length (o comprimento dos segmentos de corres nas barras) dos parâmetros são definidos nos parâmetros externos do indicador para a possibilidade de configuração manual (a guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Um indicador para demonstrar RAW_COLOR_LINE"
#property description "Ele desenha uma linha no Fechamento do preço em partes colorida
#property description "A largura, estilo e cor de partes da linha são alteradas aleato
#property description "cada N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- Define 5 cores para colorir cada barra (elas são armazenadas no array especial)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Podem se
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;           //Número de ticks para alterar
input int      Length=20;     // A largura de cada parte de cor nas barras
int            line_colors=5; // O número para definir as cores é 5 - ver #property in
//--- Um buffer para plotagem
double         ColorLineBuffer[];
//--- Um buffer para armazenar a cor da linha sobre cada barra
double         ColorLineColors[];

//--- O array para armazenar cores contendo 7 elementos
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Ligando um array e um buffer de indicador
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- Inicializando o gerador de números pseudo-aleatórios
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Alterar as cores de seções de linhas
      ChangeColors(colors,5);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Bloco para cálculo dos valores do indicador
   for(int i=0;i<rates_total;i++)
     {
      //--- Escrever o valor do indicador para o buffer
      ColorLineBuffer[i]=close[i];
      //--- Agora, aleatoriamente definir um índice de cores para esta barra
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- Para esta barra, a linha terá a cor com o índice color_index
      ColorLineColors[i]=color_index;
     }

//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
```

```
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência de uma linha visualizada no indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+" Width="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   int size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_SECTION

O estilo DRAW_COLOR_SECTION é uma versão de cor de DRAW_SECTION, mas ao contrário deste último, permite seções de desenho de cores diferentes. O estilo DRAW_COLOR_SECTION, como todos os estilos de cor com a palavra **COLOR** em seu título tem um buffer de indicador adicional e especial que armazena o índice de cores (número) a partir de um conjunto especial array de cores. Assim, a cor de cada uma das seções pode ser definido através da especificação do índice de cores do índice da barra que corresponde a seção final.

A largura, cor e estilo das seções podem ser especificados como para o estilo DRAW_SECTION - usando as as diretivas do compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas aparências sejam alteradas dependendo da situação corrente.

Seções são desenhadas a partir de um valor não vazio para outro valor não vazio do buffer de indicador, valores vazios são ignorados. Para especificar qual o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE: Por exemplo, se o indicador deve ser desenhado como uma seqüência de seções sobre valores diferentes de zero, então você precisa definir o valor zero como um vazio:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

Sempre preencha explicitamente nos valores dos buffers do indicador, defina um valor vazio num buffer para os elementos que não devem ser plotados.

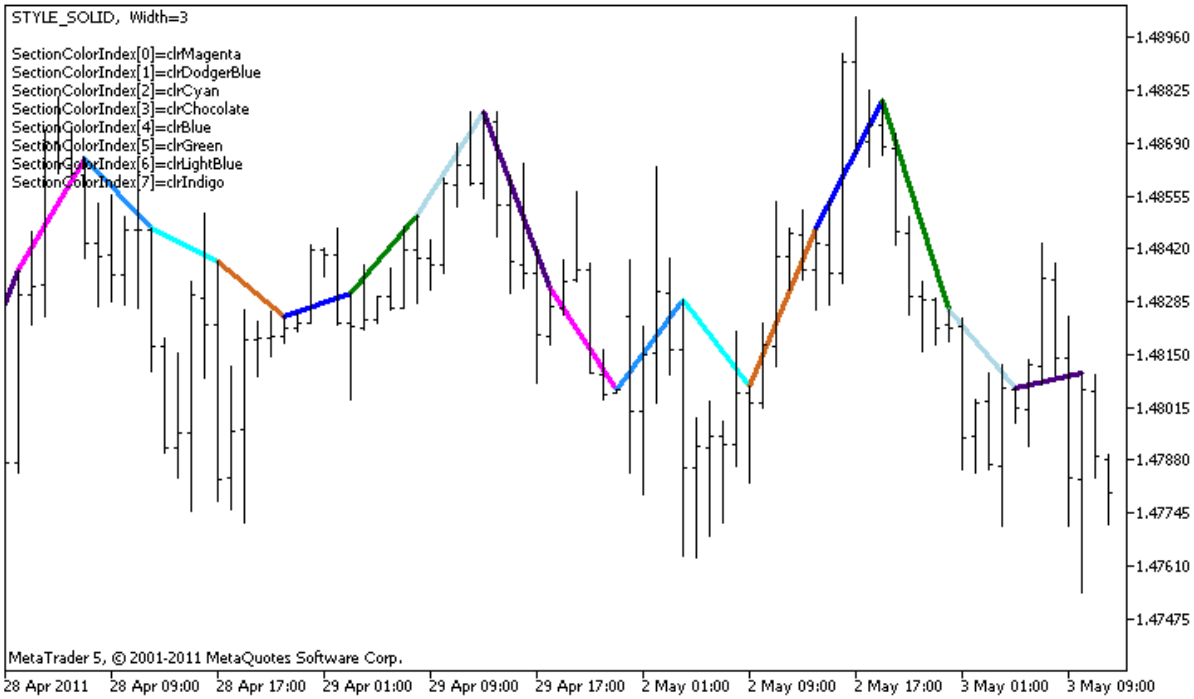
O número de buffers requirido para plotagem de DRAW_COLOR_SECTION é 2.

- um buffer para armazenar os valores do indicador utilizado para desenhar uma linha;
- um buffer para armazenar o índice de cor, que é usada para desenhar a seção (faz sentido definir apenas valores não vazios).

Cores serão especificadas pela diretiva de compilador #property indicator_color1 separada por uma vírgula. O número de cores não pode exceder a 64.

Um exemplo do indicador que desenha seções coloridas a cada 5 barras longas, usando os valores da alta do preço. A cor, a largura e o estilo das seções alteram aleatoriamente a cada **N** ticks.

© 2000-2024, MetaQuotes Ltd.



Note que para inicializar **plot1** com DRAW_COLOR_SECTION, 8 cores são definidas usando a diretiva de compilador #property. Então na função OnCalculate(), cores são definidas aleatoriamente a partir do array de cores colors[].

O parâmetro N é defindo nos parâmetros externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_SECTION"
#property description "Ele desenha secções coloridas com o comprimento igual ao número
#property description "A cor, a largura eo estilo de seções são alteradas aleatoriamen
#property description "depois de cada N ticks"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- Definir 8 cores para colorir seções (eles são armazenados numa array especial)
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,
#property indicator_style1  STYLE_SOLID
```

```
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;                      // Número de ticks para alterar
input int      bars_in_section=5;        // O comprimento das seções nas barras
//--- Uma variável auxiliar para calcular extremidades das seções
int            divider;
int            color_sections;
//--- Um buffer para plotagem
double         ColorSectionBuffer[];
//--- Um buffer para armazenar a cor da linha sobre cada barra
double         ColorSectionColors[];
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- O número de cores para colorir as secções
   int color_sections=8;   //  veja um comentário para #property indicator_color1
//--- Verifica o parâmetro do indicador
   if(bars_in_section<=0)
     {
      PrintFormat("Comprimento seção inválido=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Alterar cores utilizadas para plotar as seções
      ChangeColors(colors,color_sections);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- O número de barra a partir da qual o cálculo dos valores do indicador inicia
   int start=0;
//--- Se o indicador foi calculado antes, então definir o início na barra anterior
   if(prev_calculated>0) start=prev_calculated-1;
//--- Aqui estão todos os cálculos dos valores do indicador
   for(int i=start;i<rates_total;i++)
     {
      //--- Se o número de barras é divisível pela section_length, isso significa que
      if(i%bars_in_section==0)
        {
         //--- Definir o final da seção na alta de preço desta barra
         ColorSectionBuffer[i]=high[i];
         //--- O resto da divisão do número de barra pela scetion_length*number_of_co
         int rest=i%divider;
         //Obter o número da cor =  a partir de 0 para number_of_colors-1
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //---Se o resto da divisão é igual para barras,
      else
        {
         //--- Se nada aconteceu, ignore a barra - definir 0
         else ColorSectionBuffer[i]=0;
        }
     }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência de uma linha visualizada no indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+" Width="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   int size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

O estilo DRAW_COLOR_HISTOGRAM desenha um histograma como uma seqüência de colunas coloridas a partir de zero pra um valor específico. Os valores são extraídos do buffer do indicador. Cada coluna pode ter a sua própria cor a partir de um conjunto pré-definido de cores.

A largura, cor e estilo do histograma podem ser especificados como para o estilo DRAW_HISTOGRAM - usando as diretivas de compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem alterar o visual do histograma com base na situação atual.

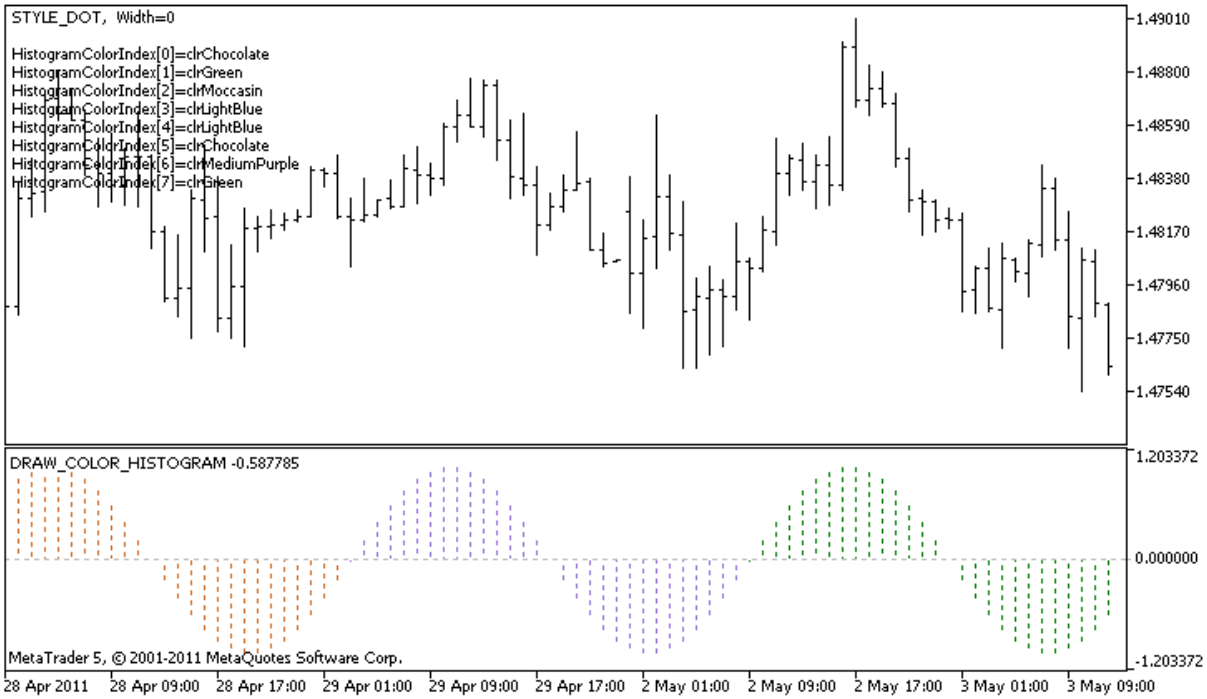
Uma vez que uma coluna a partir do nível zero é desenhado em cada barra, DRAW_COLOR_HISTOGRAM será melhor utilizado numa janela de gráfico em separado. Na maioria das vezes este tipo de plotagem é usada para criar indicadores do tipo osciladores, por exemplo, Awesome Oscillator ou Market Facilitation Index. Para os valores vazios que não são visíveis, o valor zero deve ser especificado.

O número de buffers necessários para plotagem DRAW_COLOR_HISTOGRAM é 2.

- um buffer para armazenar um valor que não seja zero do segmento vertical em cada barra, a segunda extremidade do segmento estará sempre na linha zero do indicador;
- um buffer para armazenar o índice de cor, que é usada para desenhar a seção (faz sentido definir apenas valores não vazios).

Cores podem ser determinadas usando a diretiva de compilador #property indicator_color1 separadas por uma vírgula. O número de cores não pode exceder a 64.

Um exemplo do indicador que desenha uma senóide de uma cor específica com base na função MathSin(). A cor, a largura e o estilo de **todas** colunas do histograma mudam aleatoriamente a cada **N** ticks. O parâmetro de barras determina o período da senóide, isto é, após um número especifico de barras a senóide vai repetir o ciclo.



© 2000-2024, MetaQuotes Ltd.

Por favor note que para **plot1** com o estilo DRAW_COLOR_HISTOGRAM, 5 cores são definidas usando a diretiva de compilador #property, e então na função OnCalculate(), as cores são selecionadas aleatoriamente a partir de quatorze cores armazenada no array de colors[]. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_HISTOGRAM"
#property description "Ele desenha um senóide como um histograma em uma janela separad
#property description "A cor e a largura das colunas são alterados aleatoriamente"
#property description "depois de cada N ticks"
#property description "O parâmetro de barras define o número de barras para repetir a

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- parâmetros de entrada
input int      bars=30;          // O período de uma senóide em barras
input int      N=5;              // O número de ticks para alterar o histograma
//--- plotar Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- Definir 8 cores para colorir seções (eles são armazenados numa array especial)
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Um buffer de valores
double         Color_HistogramBuffer[];
//--- Um buffer de índices de cor
double         Color_HistogramColors[];
//--- Um fator para obter o ângulo 2Pi em radianos, quando multiplicado pelo parâmetro
double         multiplier;
int            color_sections;
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
```

```
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- O número de cores para a colorir a senóide
   color_sections=8;   //  Veja comentário para #property indicator_color1
//--- Calcular a multiplicação
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Defina o valor das barras=%d maior do que 1",bars);
      //--- Finalização antecipada do indicador
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Alterar cores usadas para o histograma
      ChangeColors(colors,color_sections);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Calcule os valores do indicador
   int start=0;
//--- Se já foi calculado durante os inícios anteriores do OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // definir o início do cálculo com a
//--- Preencher o buffer do indicador com valores
   for(int i=start;i<rates_total;i++)
     {
      //--- Um valor
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- Cor
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência de uma linha visualizada no indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+" Width="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   int size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

O estilo DRAW_COLOR_HISTOGRAM2 desenha uma histograma de uma cor específica – segmentos verticais usando os valores de buffers do indicador. Mas diferentemente ao de uma cor DRAW_HISTOGRAM2, este estilo pode ter a sua própria cor a partir de um conjunto pré-definido em cada coluna do histograma. Os valores de todos os segmentos são retirados do buffer do indicador.

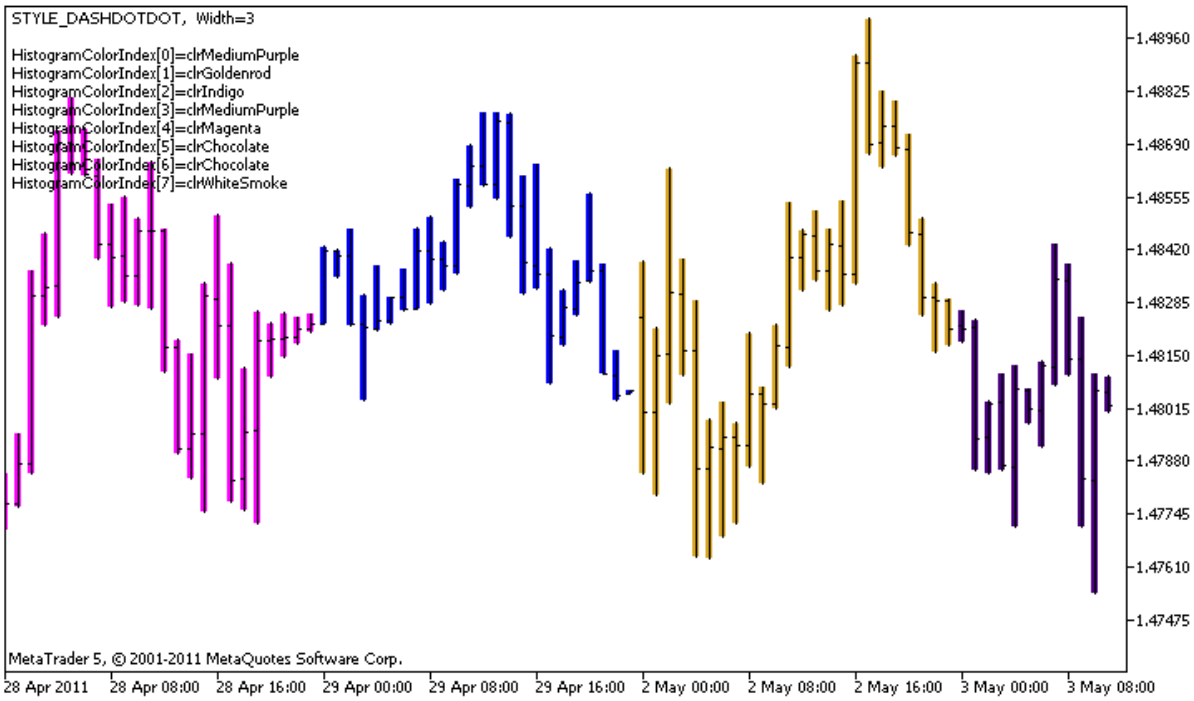
A largura, o estilo e a cor do histograma pode ser especificado como para o estilo DRAW_HISTOGRAM2 – usando as diretivas de compilador ou usando dinamicamente a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem alterar o visual do histograma com base na situação atual.

O estilo DRAW_COLOR_HISTOGRAM2 pode ser usado em uma sub-janela separada de um gráfico e em sua janela principal. Para valores vazios nada é desenhado, todos os valores nos buffers do indicador precisam ser definidos explicitamente. Buffers não são inicializados com valores vazios.

O número de buffers necessários para plotagem do DRAW_COLOR_HISTOGRAM2 é 3:

- dois buffers para armazenar a extremidade superior e inferior do segmento vertical em cada barra;
- um buffer para armazenar o índice de cor, o qual é usado para desenhar o segmento (faz sentido definir apenas valores não vazios).

Um exemplo de indicador que desenha um histograma de uma cor especifica entre os preços de alta e baixa. Para cada dia da semana, as linhas de histograma ter uma cor diferente. A cor do dia, a largura e o estilo do histograma é alterado aleatoriamente a cada **N** ticks.



Por favor note que para **plot1** com o estilo DRAW_COLOR_HISTOGRAM2, 5 cores são definidas usando a diretiva de compilador #property, e então na função OnCalculate() as cores são selecionadas aleatoriamente a partir de 14 cores armazenadas no array colors[].

O parâmetro N é defindo nos parâmetros externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM2.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_HISTOGRAM2"
#property description "É desenhado um segmento entre a abertura e fechamento em cada k
#property description "A cor, a largura e o estilo são alterados aleatoriamente"
#property description "depois de cada N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plotar ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- Definir 5 cores para colorir o histograma com base nos dias de semana (elas são
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parâmetros de entrada
input int      N=5;              // O número de ticks para alterar o histograma
int            color_sections;
//--- Valores de buffers
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- Um buffer de índices de cor
double         ColorHistogram_2Colors[];
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
```

```
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- Definir um valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- O número de cores para a colorir a senóide
   color_sections=8;   //  Veja um comentário #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Mude as cores usadas para desenhar o histograma
      ChangeColors(colors,color_sections);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Calcule os valores do indicador
   int start=0;
//--- Para obter o dia da semana pela abertura de preço de cada barra
   MqlDateTime dt;
//--- Se já foi calculado durante os inícios anteriores do OnCalculate
   if(prev_calculated>0) start=prev_calculated-1; // definir o início do cálculo com a
//--- Preencher o buffer do indicador com valores
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      //--- valor
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Define o índice de cores de acordo com o dia da semana
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- Retorna o valor de prev_calculated para a próxima chamada da função
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência de uma linha visualizada no indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
```

© 2000-2024, MetaQuotes Ltd.

```
   comm=comm+" Width="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   int size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_ARROW

O estilo DRAW_COLOR_ARROW desenha setas coloridas (símbolos do conjunto Wingdings) baseado no valor do buffer do indicador. Em contraste com DRAW_ARROW, neste estilo, é possível definir uma cor partir de um conjunto predefinido de cores especificada pela propriedade indicator_color1 para cada símbolo.

A largura e a cor dos símbolos podem ser especificadas com o estilo DRAW_ARROW - usando diretivas de compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem modificar a aparência de um indicador com base na situação atual.

O código de símbolo é definido usando a propriedade PLOT_ARROW.

```
//--- Definir o código símbolo a partir das fontes Wingdings para desenhar em PLOT_ARR
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

O valor padrão do PLOT_ARROW=159 (um círculo).

Cada seta é na verdade um símbolo que tem a altura e o ponto de ancoragem e pode abranger algumas informações importantes num gráfico (por exemplo, o preço de fechamento da barra). Assim, podemos adicionalmente especificar o deslocamento vertical em pixels, o qual não depende da escala do gráfico. As setas serão deslocadas para baixo pelo número especificado de pixels, embora os valores do indicador permanecerão os mesmos:

```
//--- Definir o deslocamento vertical das setas em pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

Um valor negativo de PLOT_ARROW_SHIFT significa a mudança de setas para cima, valores positivos deslocam a seta para baixo.

O estilo DRAW_COLOR_ARROW pode ser usado em separado na sub-janela do gráfico e em sua janela principal. Valores vazios não são desenhados e não aparecem na "Janela de Dados", todos os valores nos buffers do indicador devem ser definidos explicitamente. Buffers não são inicializados com o valor zero.

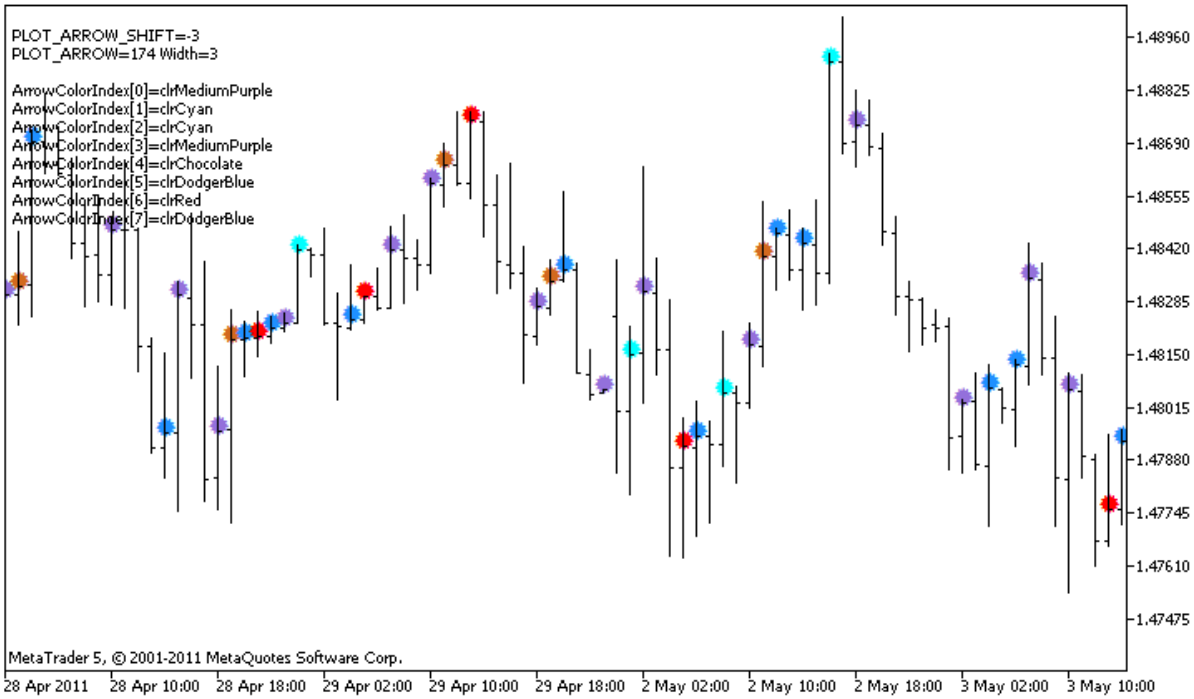
```
//--- Definir um valor vazio
   PlotIndexSetDouble(plot_index_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

O número de buffers necessários para plotagem de DRAW_COLOR_ARROW é 2.

- Um buffer para armazenar o valor do preço que é utilizado para desenhar o símbolo (além de um deslocamento em pixels, determinado na propriedade PLOT_ARROW_SHIFT);
- Um buffer para guardar o índice de cor que é utilizado para desenhar uma seta(faz sentido configurar apenas valores não vazios).

Um exemplo de indicador que desenha as setas em cada uma das barras com o preço mais alto do que o preço final da barra anterior. A largura, deslocamento e símbolo de código de **todas** setas são alterados aleatoriamente a cada **N** ticks. A cor do símbolo depende do número da barra na qual está desenhada.

© 2000-2024, MetaQuotes Ltd.



Neste exemplo, para **plot1** com o estilo DRAW_COLOR_ARROW, as propriedades, cor e tamanho são especificadas usando a diretiva de compilador #property, e então, na função OnCalculate() as propriedades são definidas aleatoriamente. O parâmetro N é definido nos parâmetros externos do indicador para a possibilidade de configuração manual (o guia Parâmetros na janela Propriedades do indicador).

Por favor, note que inicialmente oito cores são usadas na diretiva de compilador #property, e então, na função OnCalculate(), a cor é definida aleatoriamente a partir das 14 cores que são armazenadas no array colors[].

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_ARROW"
#property description "Desenha setas de diferentes cores definidas pelo caracteres Uni
#property description "A cor, o tamanho, deslocamento e código de símbolo da seta são
#property description " aleatoriamente a cada N ticks"
#property description "O parâmetro de código define o valor da base: código = 159 (um

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar ColorArrow
#property indicator_label1  "ColorArrow"
```

```
#property indicator_type1   DRAW_COLOR_ARROW
//--- Define 8 cores para colorir o histograma com base nos dias de semana (eles são a
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- parâmetros de entrada
input int      N=5;         // Número de ticks para mudar
input ushort   code=159;    // Código símbolo para desenhar em DRAW_ARROW
int            color_sections;
//--- Um buffer do indicador para plotar
double         ColorArrowBuffer[];
//--- Um buffer para armazenar índices de cor
double         ColorArrowColors[];
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- Definir o código símbolo para desenho em PLOT_ARROW
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Definir o deslocamento vertical das setas em pixels
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Definir como um valor vazio 0
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- O número de cores para a colorir a senóide
   color_sections=8;   //  veja um comentário #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcular ticks para mudar a cor, tamanho, deslocamento e código da seta
   ticks++;
//--- Se um número crítico de ticks tem sido acumulado
   if(ticks>=N)
     {
      //--- Alterar propriedades de seta
      ChangeLineAppearance();
      //--- Mude as cores usadas para desenhar o histograma
      ChangeColors(colors,color_sections);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- Bloco para cálculo dos valores do indicador
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- Calculando loop
   for(int i=1;i<rates_total;i++)
     {
      //--- Se o preço atual fechado é mais alto do que o anterior, desenhar uma seta
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- Caso contrário, especificar o valor nulo
      else
         ColorArrowBuffer[i]=0;
      //--- Cor da seta
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Altera a cor dos segmentos de linha                              |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- Obter um valor aleatório
       int number=MathRand();
       //--- Obter um índice no array col[] como um resto da divisão inteira
       int i=number%size;
       //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
       PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                           PLOT_LINE_COLOR,      //  Identificador da propriedade
                           plot_color_ind,       //  O índice da cor, onde se escreve a
                           cols[i]);             //  Uma nova cor
       //--- Escrever as cores
       comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
       ChartSetString(0,CHART_COMMENT,comm);
      }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência de uma linha visualizada no indicador         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades de linha
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5; // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+" Width="+IntegerToString(width);

//--- Um bloco para mudar o código das setas (PLOT_ARROW)
   number=MathRand();
//--- Obter o restante da divisão de número inteiro para calcular um novo código da se
   int code_add=number%20;
//--- Definir o novo símbolo de código como o resultado de code+code_add
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- Escreva o código de símbolo PLOT_ARROW
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Um bloco de mudança do deslocamento vertical das setas em pixels
   number=MathRand();
//--- Obter o deslocamento como o restante da divisão inteira
   int shift=20-number%41;
//--- Define um deslocamento a partir de
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- Escrever o deslocamento PLOT_ARROW_SHIFT
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

O estilo DRAW_COLOR_ZIGZAG desenha segmentos cores diferentes, usando os valores de dois buffers do indicador. Este estilo é uma versão colorida de DRAW_ZIGZAG, ou seja, permite a especificar para cada segmento uma cor individual a partir do conjunto pré-definido de cores. Os segmentos são plotados a partir de um valor no primeiro buffer para um valor no segundo buffer do indicador. Nenhum dos buffers pode conter somente valores vazios, uma vez que neste caso nada é plotado.

A largura, cor e estilo das linhas podem ser especificadas como para o estilo DRAW_ZIGZAG - usando as diretivas de compilação ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas aparências sejam alteradas dependendo da situação corrente.

Seções são desenhadas a partir de um valor não vazio de um buffer para um valor não vazio de outro buffer do indicador. Para especificar o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE:

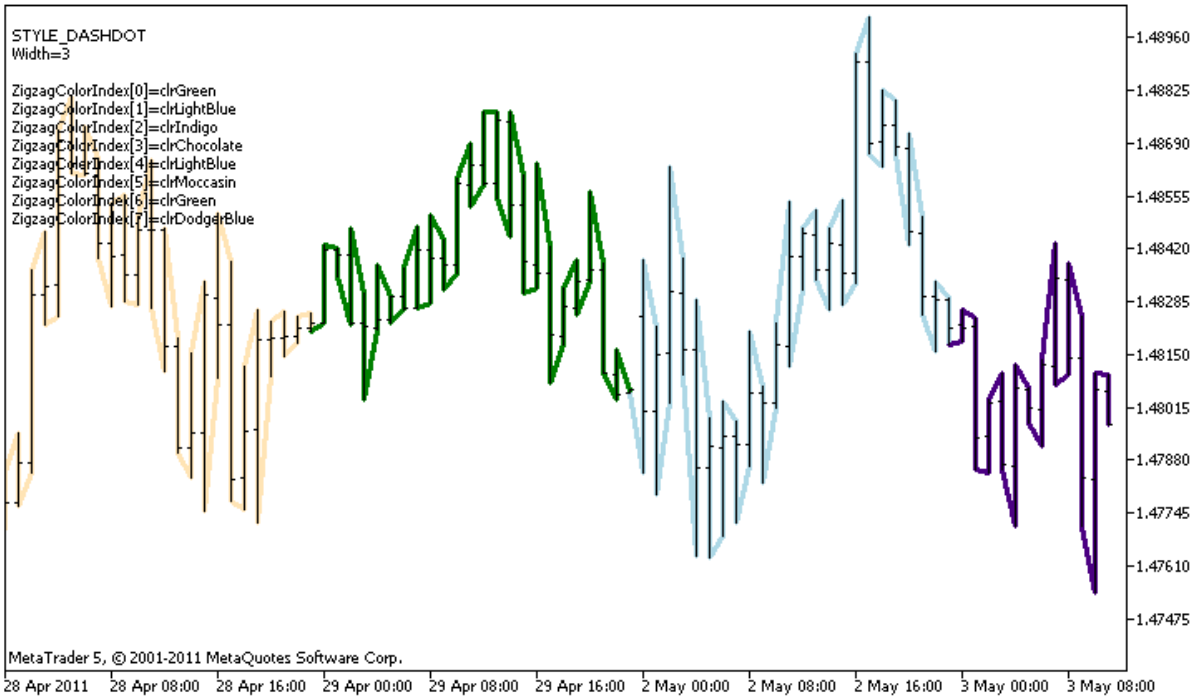
```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Sempre preencha explicitamente nos valores dos buffers de indicador, defina um valor vazio num buffer para saltar barras.

O número de buffers requerida para plotagem de DRAW_COLOR_ZIGZAG é 3:

- dois buffers para armazenar os valores das extremidades das seções em zigue-zague;
- um buffer para armazenar o índice de cor, que é usada para desenhar a seção (faz sentido definir apenas valores não vazios).

Um exemplo de indicador de que plota uma serra com base nos altas e baixas dos preços. A cor, a largura e o estilo das linhas em ziguezague alteram aleatoriamente a cada **N** ticks.



© 2000-2024, MetaQuotes Ltd.

Por favor note que para **plot1** com o estilo DRAW_COLOR_ZIGZAG, 8 cores são definidas usando a diretiva de compilação #property, e então na função OnCalculate() a cor é selecionada aleatoriamente a partir de 14 cores armazenadas no array colors[].

O parâmetro N é defindo nos parâmetros externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_ZIGZAG"
#property description "Ele desenha uma linha tracejada, como uma seqüência das seções
#property description "A cor, a largura e tamanho dos segmentos são alterados aleator
#property description " a cada N ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- plotar Color_Zigzag
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- Definir 8 cores para colorir seções (eles são armazenados numa array especial)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int      N=5;              // Número de ticks para alterar
int            color_sections;
//--- Buffers dos valores das extremidades do segmento
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- Buffers do índice de cores das extremidades dos segmento
double         Color_ZigzagColors[];
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPur
  };
//--- Um array para armazenar os estilos de linha
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//----Número de corres para colorir o zigzag
   color_sections=8;   //  veja um comentário para a #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Calcule ticks para mudar o estilo, cor e largura da linha
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Alterar cores utilizadas para plotar as seções
      ChangeColors(colors,color_sections);
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }

//--- A estrutura de tempo é necessário para obter o dia da semana de cada barra
   MqlDateTime dt;

//--- A posição inicial dos cálculos
   int start=0;
//--- Se o indicador foi calculado com o tick anterior, então inicia o cálculo com o p
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Calculando loop
   for(int i=start;i<rates_total;i++)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- Escreva o tempo de abertura da barra na estrutura
       TimeToStruct(time[i],dt);

       //--- Se é ainda o número de barra
       if(i%2==0)
         {
          //---  Escrever a alta no primeiro buffer e a baixa no segundo
          Color_ZigzagBuffer1[i]=high[i];
          Color_ZigzagBuffer2[i]=low[i];
          //--- A cor do segmento
          Color_ZigzagColors[i]=dt.day_of_year%color_sections;
         }
       //--- o número da barra é ímpar
       else
         {
          //--- Preencha a barra numa ordem inversa
          Color_ZigzagBuffer1[i]=low[i];
          Color_ZigzagBuffer2[i]=high[i];
          //--- A cor do segmento
          Color_ZigzagColors[i]=dt.day_of_year%color_sections;
         }
      }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Alterar a cor do segmento zigzag                                 |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
```

```
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Alterar a aparência dos segmentos zigzag                         |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação da informação sobre as propriedades de Color_ZigZag
   string comm="";
//--- Um bloco para alterar a largura da linha
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Um bloco para mudar o estilo da linha
   number=MathRand();
//--- O divisor é igual ao tamanho do array de estilos
   int size=ArraySize(styles);
//--- Obter o índice para selecionar um novo estilo como o resto da divisão inteira
   int style_index=number%size;
//--- Definir a cor com a propriedade PLOT_LINE_COLOR
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Escrever o estilo de linha
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_BARS

O estilo DRAW_COLOR_BARS desenha barra sobre valores de quatro buffers de indicador, que contém Open (abertura), High (alta), Low (baixa) e Close (fechamento) de preços. Este estilo é uma versão avançada do DRAW_BARS e permite a especificação para cada barra de uma cor individual do conjunto predefinido de cores. É usado para a criação de indicadores personalizados como barras, incluindo aqueles numa sub-janela separada de um gráfico e sobre outros instrumentos financeiros.

A cor das barras pode ser definida usando as diretivas de compilador ou dinamicamente utilizando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O indicador é desenhado apenas para as barras, para o qual os valores não vazios de **todos** quatro buffers do indicador são definidos. Para especificar o valor que deve ser considerado como "vazio", defina este valor na propriedade PLOT_EMPTY_VALUE:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

Sempre preencher explicitamente os valores dos buffers do indicador, definir um valor vazio num buffer para ignorar barras.

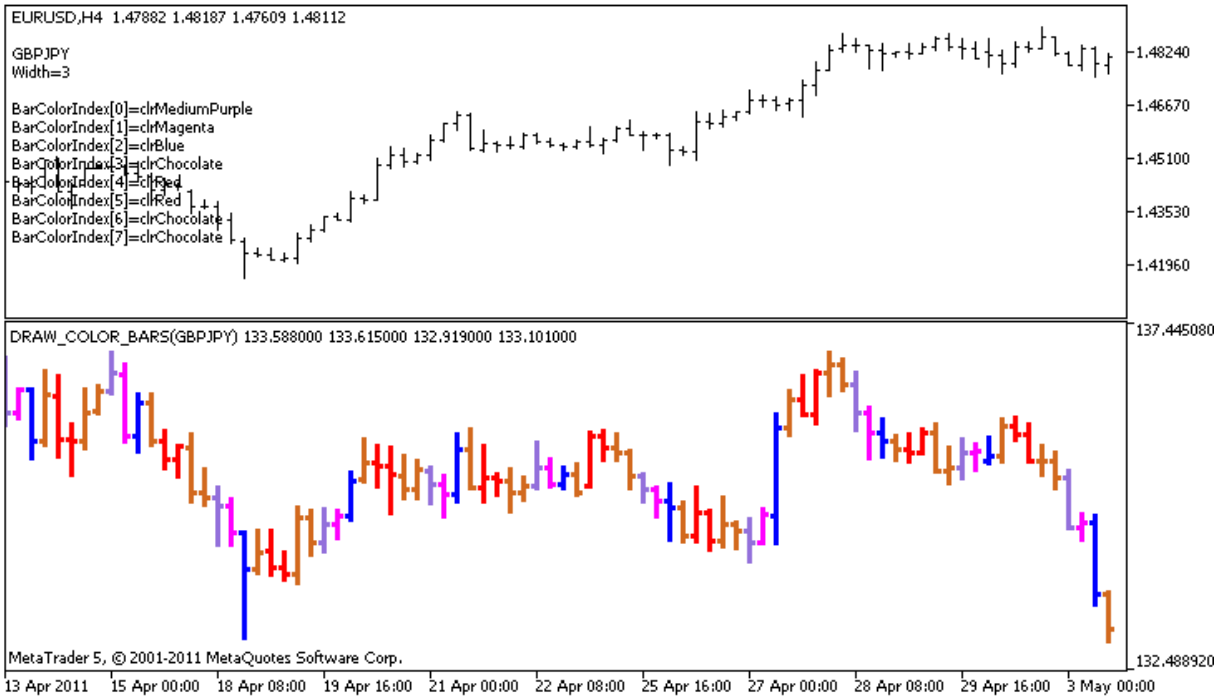
O número requerido de buffers para plotagem de DRAW_COLOR_BARS é 5:

- quatro buffers para armazenar Open, High, Low e Close;
- um buffer para armazenar o índice das cores, que é usado para desenhar uma barra(faz sentido defini-lo apenas para as barras que serão desenhadas).

Todos os buffers para a plotagem devem ir um após o outro na ordem determinada: Open, High, Low, Close e o buffer de cor. Nenhum dos buffers de preços podem conter valores nulos, uma vez que neste caso nada é plotado.

Um exemplo do indicador que desenha barras sobre um instrumento financeiro selecionado numa janela separada. A cor das barras muda aleatoriamente a cada **N** ticks. O parâmetro N é definido nos parâmetro externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

© 2000-2024, MetaQuotes Ltd.



Por favor note que para **plot1** com o estilo DRAW_COLOR_BARS, 8 cores são definidas usando a diretiva de compilador #property, e então na função OnCalculate() a cor é selecionada aleatoriamente a partir de 14 cores armazenadas no array de colors[].

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                          Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_BARS"
#property description "Ele desenha diferentes cores nas barras de um símbolo selecion
#property description "A cor e a largura das barras, bem como o símbolo são alterados
#property description "cada N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plotar ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- Define 8 cores para colorir barras (elas são armazenadas num array especial)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int       N=5;              // O número de ticks para alterar o tipo
```

```
input int      bars=500;         // O número de barras para mostrar
input bool     messages=false;   // Mostrar mensagens no log "Expert Advisors"
//--- buffers do Indicador
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- Nome do símbolo
string symbol;
int    bars_colors;
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- Número de cores para colorir as barras
   bars_colors=8;   //  veja uma comentário para a #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- Conte ticks para mudar o estilo, cor e largura da barra
```

© 2000-2024, MetaQuotes Ltd.

```
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Selecione um novo símbolo a partir da janela de observação do Mercado
      symbol=GetRandomSymbolName();
      //--- Alterar as propriedades da linha
      ChangeLineAppearance();
      //--- Mude as cores usadas para desenhar os candlesticks
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- Faça 5 tentativas de preencher os buffers com os preços a partir símbolo
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- Um contador de chamadas da função CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher os buffers do indicador com preços                     |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- No array de rates[], vamos copiar Open (abertura), High (alta), Low (baixa) e C
   MqlRates rates[];
//--- O contador de tentativas
   int attempts=0;
//--- Quanto foi copiado
   int copied=0;
//--- Fazer 25 tentativas para obter uma série temporal(timeseries) sobre o símbolo de
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s Copiar taxas(%s) tentativas=%d",__FUNCTION__,nome,t
     }
//--- Se falhou em copiar um número suficiente de barras
   if(copied!=bars)
     {
      //--- Formar uma mensagem string
      string comm=StringFormat("Para o símbolo %s, conseguiu receber somente %d barras
                               name,
                               copied,
                               bars
```

```
                                 );
      //--- Mostrar uma mensagem num comentário na janela do gráfico principal
      Comment(comm);
      //--- Mostrar a mensagem
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Definir a exibição do símbolo
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- Inicializar buffers com valores vazios
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- Copiar os preços para os buffers
   for(int i=0;i<copied;i++)
     {
      //--- Calcular o índice apropriado para os buffers
      int buffer_index=total-copied+i;
      //--- Escreva os preços para os buffers
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Aleatoriamente retorna um símbolo de Observação do Mercado       |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- O número de símbolos mostrada na janela de observação do Mercado
   int symbols=SymbolsTotal(true);
//--- A posição de um símbolo na lista - um número aleatório de 0 para símbolos
   int number=MathRand()%symbols;
//--- Retornar o nome de um símbolo na posição especificada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//|  Alterar a cor do segmento zigzag                                |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Altera a aparência das barras                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades da barra
   string comm="";

//--- Um bloco para alterar a largura das barras
   int number=MathRand();
//--- Obter a largura do restante da divisão inteira
   int width=number%5;   // A largura é definida de 0 a 4
//--- Definir a cor com a propriedade PLOT_LINE_WIDTH
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Escrever a largura da linha
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Escreve o nome do símbolo
   comm="\r\n"+symbol+comm;

//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

O estilo DRAW_COLOR_CANDLES style, como o DRAW_CANDLES, desenha candlesticks usando valores de quatro buffers de indicador o qual contém Open (abertura), High (alta), Low (baixa) e Close (fechamento) de preços. Além disso, ele permite a especificação de uma cor para cada um candlestick de um determinado conjunto. Para esta finalidade, o estilo tem um buffer de cor especial que armazena os índices de cor para cada barra. É usado para a criação de indicadores personalizados como uma seqüência de candlesticks, incluindo os inseridos numa sub-janela separada de um gráfico e noutros instrumentos financeiros.

O número de cores de candlesticks pode ser definido usando as diretivas de compilador ou dinamicamente usando a função PlotIndexSetInteger(). Mudanças dinâmicas das propriedades de plotagem permitem "avivar" os indicadores, de modo que suas mudanças de aparência dependem da situação corrente.

O indicador é desenhado apenas para as barras, para o qual os valores não vazios de quatro buffers de preços do indicador são definidos. Para especificar o valor que deve ser considerado como "vazio", defina esse valor na propriedade PLOT_EMPTY_VALUE:

```
//--- O valor 0 (vazio) não vai participar no desenho
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

Sempre preencher explicitamente os valores dos buffers do indicador, definir um valor vazio num buffer para ignorar barras.

O número de buffers necessários para plotagem de DRAW_COLOR_CANDLES é 5:

- quatro buffers para armazenar Open, High, Low e Close;
- um buffer para armazenar o índice de cor, que é usada para desenhar um candlestick (faz sentido defini-lo apenas para os candlesticks que serão desenhados).

Todos os buffers para a plotagem devem ir um após o outro na ordem determinada: Open, High, Low, Close e o buffer de cor. Nenhum dos buffers de preços pode conter apenas valores vazios, uma vez que, neste caso, nada é plotado.

Um exemplo de indicador que desenha candlesticks para um instrumento financeiro selecionado em uma janela separada. A cor dos candlesticks muda aleatoriamente a cada **N** ticks. O parâmetro N é definido nos parâmetro externos do indicador para a possibilidade de configuração manual (na guia Parâmetros na janela Propriedades do indicador).

© 2000-2024, MetaQuotes Ltd.

Por favor, note que para **plot1**, a cor é definida usando a diretiva de compilador #property, e então, na função OnCalculate() a cor é definida aleatoriamente a partir de uma lista preparada anteriormente.

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Um indicador para demonstrar DRAW_COLOR_CANDLES."
#property description "ele desenha candlesticks de um símbolo selecionado numa janela
#property description " "
#property description "A cor e a largura dos candlesticks, bem como o símbolo são alte
#property description "aleatoriamente a cada N ticks"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- Plotar ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- Definir 8 cores para colorir os candlesticks (eles são armazenados na array espe
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//--- parâmetros de entrada
input int      N=5;              // O número de ticks para alterar o tipo
input int      bars=500;         // O número de candlesticks para mostrar
input bool     messages=false;   // Mostrar mensagens no log "Expert Advisors"
//--- buffers do Indicador
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- Nome do símbolo
string symbol;
//--- Um array para armazenar as cores que contém 14 elementos
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Se as barras são muito pequenas - concluir o trabalho antes do tempo
   if(bars<50)
     {
      Comment("Por favor, especifique um número maior de barras! A operação do indicad
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- Um valor vazio
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- O nome do símbolo, para o qual as barras são desenhadas
   symbol=_Symbol;
//--- Definir a exibição do símbolo
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- O número de cores para a cor dos candlesticks
   candles_colors=8;     //  Veja uma comentário para a #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- Contar ticks para mudar o estilo e cor
   ticks++;
//--- Se um número suficiente de ticks foi acumulado
   if(ticks>=N)
     {
      //--- Selecione um novo símbolo a partir da janela de observação do Mercado
      symbol=GetRandomSymbolName();
      //--- Alterar a forma
      ChangeLineAppearance();
      //--- Mude as cores usadas para desenhar os candlesticks
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- Faça 5 tentativas de preencher os buffers de Plot1 com os preços a partir
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- Um contador de chamadas da função CopyFromSymbolToBuffers()
         tries++;
        }
      //--- Redefinir o contador de ticks para zero
      ticks=0;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenche o candlestick especificado                              |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- No array de rates[], vamos copiar Open (abertura), High (alta), Low (baixa) e Cl
   MqlRates rates[];
//--- O contador de tentativas
   int attempts=0;
//--- Quanto foi copiado
   int copied=0;
//--- Fazer 25 tentativas para obter uma série temporal(timeseries) sobre o símbolo de
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s Copiar taxas(%s) tentativas=%d",__FUNCTION__,nome,t
     }
//--- Se falhou em copiar um número suficiente de barras
   if(copied!=bars)
     {
      //--- Formar uma mensagem string
      string comm=StringFormat("Para o símbolo %s, conseguiu receber somente %d barras
                               name,
                               copied,
                               bars
                               );
      //--- Mostrar uma mensagem num comentário na janela do gráfico principal
      Comment(comm);
      //--- Mostrar a mensagem
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Definir a exibição do símbolo
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- Inicializar buffers com valores vazios
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- Sobre cada tick é copiado preços para os buffers
   for(int i=0;i<copied;i++)
     {
```

```
      //--- Calcular o índice apropriado para os buffers
      int buffer_index=total-copied+i;
      //--- Escreva os preços para os buffers
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- Definir a cor do candlestick
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Aleatoriamente retorna um símbolo de Observação do Mercado       |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- O número de símbolos mostrada na janela de observação do Mercado
   int symbols=SymbolsTotal(true);
//--- A posição de um símbolo na lista - um número aleatório de 0 para símbolos
   int number=MathRand()%symbols;
//--- Retornar o nome de um símbolo na posição especificada
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Mudar a cor dos segmentos de candlestick                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- O número de cores
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Para cada índice de cor definir uma nova cor aleatoriamente
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Obter um valor aleatório
      int number=MathRand();
      //--- Obter um índice no array col[] como um resto da divisão inteira
      int i=number%size;
      //--- Definir a cor para cada índice com a propriedade PLOT_LINE_COLOR
      PlotIndexSetInteger(0,                    //  O número do estilo gráfico
                          PLOT_LINE_COLOR,      //  Identificador da propriedade
                          plot_color_ind,       //  O índice da cor, onde se escreve a
                          cols[i]);             //  Uma nova cor
      //--- Escrever as cores
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Mudar a aparência dos candlesticks                               |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Uma string para a formação de informações sobre as propriedades do candlestick
   string comm="";
//--- Escreve o nome do símbolo
   comm="\r\n"+symbol+comm;
//--- Mostrar a informação no gráfico usando um comentário
   Comment(comm);
  }
```

© 2000-2024, MetaQuotes Ltd.

Conexão entre as propriedades de indicadores e funções correspondentes

Cada indicador personalizado tem inúmeras propriedades, algumas das quais são obrigatórias e estão sempre posicionadas no início da descrição. São as seguintes propriedades:

- indicação de uma janela para plotar o indicador - indicator_separate_window ou indicator_chart_window;
- número de buffers do indicador - indicator_buffers;
- número de plotagem do indicador - indicator_plots.

Também existem outras propriedades que podem ser definidas tanto através das diretivas pré processador como das funções destinadas à criação do indicador personalizado. Estas propriedades e funções correspondentes são descritas na seguinte tabela.

Diretivas para propriedades de sub-janela de indicador	Funções do tipo IndicatorSet...()	Descrição para ajustar propriedades da sub-janela
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	O valor fixo da altura da sub-janela
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	O valor mínimo do eixo vertical
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	O valor máximo do eixo vertical
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	Valor de eixo vertical para N nível
nenhuma diretiva de pré processador	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Nome de um nível apresentado
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	Cor de N nível
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	Largura da linha para N nível
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	Estilo de linha para N nível
Diretivas para plotagem de propriedades	**Funções do tipo PlotIndexSet...()**	**Descrição do ajuste da propriedade de plotar**

© 2000-2024, MetaQuotes Ltd.

Diretivas para propriedades de sub-janela de indicador	Funções do tipo IndicatorSet...()	Descrição para ajustar propriedades da sub-janela
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	Nome abreviado do N número de plotar. Ele é exibido numa janela de dados e na dica pop-up quando passa-se o curso sobre ele
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	Cor da linha para N plotagem
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	Estilo da linha para N plotagem
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	Tipo de linha para N plotagem
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	Largura da linha para N plotagem
Propriedades comum do indicador	**Funções do tipo IndicatorSet...()**	**Description**
nenhuma diretiva de pré processador	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Define o nome abreviado conveniente do indicador, que será exibido na lista de indicadores (aberto no terminal pressionando **Ctrl+I**).
nenhuma diretiva de pré processador	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Conjuntos necessários para precisão da exibição de valores dos indicadores - número de casas decimais
nenhuma diretiva de pré processador	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Define o número de níveis na janela do indicador
indicator_applied_price	Sem função, a propriedade pode ser definida apenas pela diretiva pré processador.	Tipo do preço padrão usado para o cálculo de indicadores. Ele é especificado quando necessário, somente quando o OnCalculate() do primeiro tipo é usado. O valor da propriedade também pode ser definido a partir do diálogo de propriedades do indicador na aba "Parâmetros" - "Aplicar para".

Deve notar-se que a numeração dos níveis e plotagens em termos de pré processador inicia-se com um, enquanto a numeração das mesmas propriedades usando funções inicia-se com zero, ou seja, o valor indicado deve ser pelo menos um do que o indicado para #property.

© 2000-2024, MetaQuotes Ltd.

Existem várias diretivas, para a qual não existem funções correspondentes:

Diretiva	Description
indicator_chart_window	Indicador é visualizado na janela principal
indicator_separate_window	Indicador é visualizado numa sub-janela separada
indicator_buffers	Número necessários de buffers do indicador
indicator_plots	Números de plotagem no indicador

© 2000-2024, MetaQuotes Ltd.

SetIndexBuffer

A função vincula um buffer específico de indicador com um array unidimensional dinâmico do tipo double.

```
bool  SetIndexBuffer(
   int                    index,          // índice de buffer
   double                 buffer[],       // array
   ENUM_INDEXBUFFER_TYPE  data_type       // O que será o armazenado
   );
```

Parâmetros

index

[in] Número do buffer do indicador. A numeração inicia com 0. O número deve ser inferior ao valor declarado em #property indicator_buffers.

buffer[]

[in] Um array declarado no programa de indicador personalizado.

data_type

[in] Tipo de dados armazenados no array do indicador. Por padrão ele é INDICATOR_DATA (valores de cálculos do indicador). Ele também pode ter o valor de INDICATOR_COLOR_INDEX; neste caso, este buffer é usado para armazenar índices de cor para o buffer do indicador anterior. Você pode especificar até cores na linha #property indicator_colorN. O valor INDICATOR_CALCULATIONS significa que o buffer é usado nos cálculos intermediário do indicador, e não se destina ao desenho.

Valor do Retorno

Com sucesso, retorna true, do contrário - false.

Observação

Após a vinculação, o array buffer[] dinâmico será indexado como os arrays comuns, mesmo que a indexação de timeseries seja obrigatoriamente pré-instalado para o array. Se você quiser alterar a ordem de acesso aos elementos do array do indicador, use a função ArraySetAsSeries() **após a vinculação do array usando a função SetIndexBuffer()**. Por favor, note que você não pode alterar o tamanho dos arrays dinâmicos definidos como buffers do indicador pela função SetIndexBuffer(). Para buffers do indicador, todas as operações de alteração de tamanho são realizadas pela execução do sub-sistema do terminal.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              TestCopyBuffer1.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plotar MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int                 shift=0;
//--- buffers do indicador
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("Buffer do indicador é timeseries = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Buffer do indicador após SetIndexBuffer() é timeseries = ",
         ArrayGetAsSeries(MABuffer));

//--- alterar a ordem dos elementos de acesso do buffer do indicador
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

© 2000-2024, MetaQuotes Ltd.

```
                const long &volume[],
                const int &spread[])
  {
//--- Copiar os valores da média móvel no buffer MABuffer
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// Dependendo dos valores AsSeries
                                       // Receberá um valor mais antigo
                                       // Ou para a barra atual inacabada

//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Também Veja

Propriedades dos indicadores personalizados, Acesso para indicadores e timeseries

© 2000-2024, MetaQuotes Ltd.

IndicatorSetDouble

A função define o valor da propriedade do indicador correspondente. Propriedade do indicador deve ser do tipo double. Existem duas variantes da função.

Chamar especificando o identificador da propriedade.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificador
   double  prop_value         // valor para ser definido
   );
```

Chamar especificando o identificador e o modificador da propriedade.

```
bool  IndicatorSetDouble(
   int     prop_id,           // identificador
   int     prop_modifier,     // modificador
   double  prop_value         // valor para ser definido
   )
```

Parâmetros

prop_id

[in] Identificador da propriedade do indicador. O valor pode ser um dos valores do enumerador ENUM_CUSTOMIND_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador da propriedade específica. Somente as propriedades de nível requerem um modificador. A numeração dos níveis a partir do 0. Isso significa que a fim de definir a propriedade para o segundo nível, você precisa determinar 1 (menor que 1 quando usar a diretiva de compilação).

prop_value

[in] Valor de propriedade.

Valor do Retorno

Em caso de execução bem sucedida, retorna true, do contrário - false.

Observação

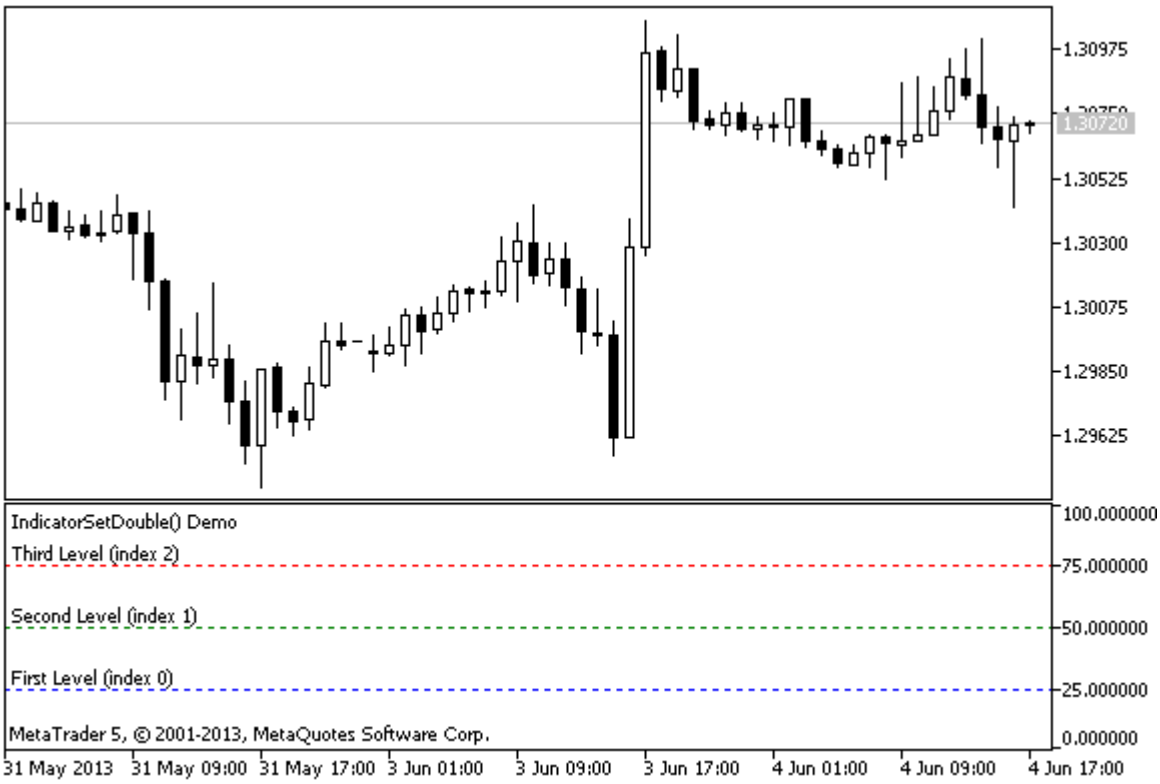
Numeração das propriedades (modificadores) começa a partir de 1 (um) quando usa a diretiva de #property, enquanto que a função usa numeração a partir de (zero). No caso do número de nível ser definido incorretamente, o visor do indicador pode ser diferente do pretendido.

Por exemplo, o primeiro valor para o indicador de nível de sub-janela separada pode ser ajustado de duas maneiras:

- property indicator_level**1** 50 - o valor 1 é utilizado para especificar o número de nível,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - 0 é usado para especificar o primeiro nível.

Exemplo: indicador inverte para baixo os valores máximo e mínimo da janela do indicador e os valores de níveis em que as linhas horizontais são colocadas.

© 2000-2024, MetaQuotes Ltd.




```
#property indicator_separate_window
//--- define o valor máximo e mínimo para a janela do indicador
#property indicator_minimum  0
#property indicator_maximum  100
//--- visualiza três níveis horizontais na janela separada
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- definir a espessura dos níveis horizontais
#property indicator_levelwidth 1
//--- definir o estilo dos níveis horizontais
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- definir as descrições dos níveis horizontais
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primeiro Nível (índice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Segundo Nível (índice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terceiro Nível (índice 2)");
//--- define o mome abreviado para o indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- definir a cor para cada nível
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

© 2000-2024, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- calcular ticks
   tick_counter++;
//--- inverter os níveis a cada 10 tick
   if(tick_counter%10==0)
     {
      //--- inverter sinal para os valores de nível
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- inverter sinal para os valores máximos e mínimos
      max-=shift;
      min-=shift;
      //--- inverter o valor de deslocamento
      shift=-shift;
      //--- definir novo valor para nível
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- definir novos valores máximo e mínimo na janela do indicador
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

© 2000-2024, MetaQuotes Ltd.

Exemplos de Estilos de Indicador, Conexão entre Propriedades do Indicador e Funções, Estilos de Desenho

© 2000-2024, MetaQuotes Ltd.

IndicatorSetInteger

A função define o valor da propriedade do indicador correspondente. Propriedade do indicador deve ser do tipo int ou color. Existem duas variantes da função.

Chamar especificando o identificador da propriedade.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificador
   int  prop_value         // valor a ser definido
   );
```

Chamar especificando o identificador e o modificador da propriedade.

```
bool  IndicatorSetInteger(
   int  prop_id,           // identificador
   int  prop_modifier,     // modificador
   int  prop_value         // valor a ser definido
   )
```

Parâmetros

prop_id

[in] Identificador da propriedade do indicador. O valor pode ser um dos valores do enumerador ENUM_CUSTOMIND_PROPERTY_INTEGER.

prop_modifier

[in] Modificador da propriedade específica. Somente as propriedades de nível requerem um modificador.

prop_value

[in] Valor de propriedade.

Valor do Retorno

Em caso de execução bem sucedida, retorna true, do contrário - false.

Observação

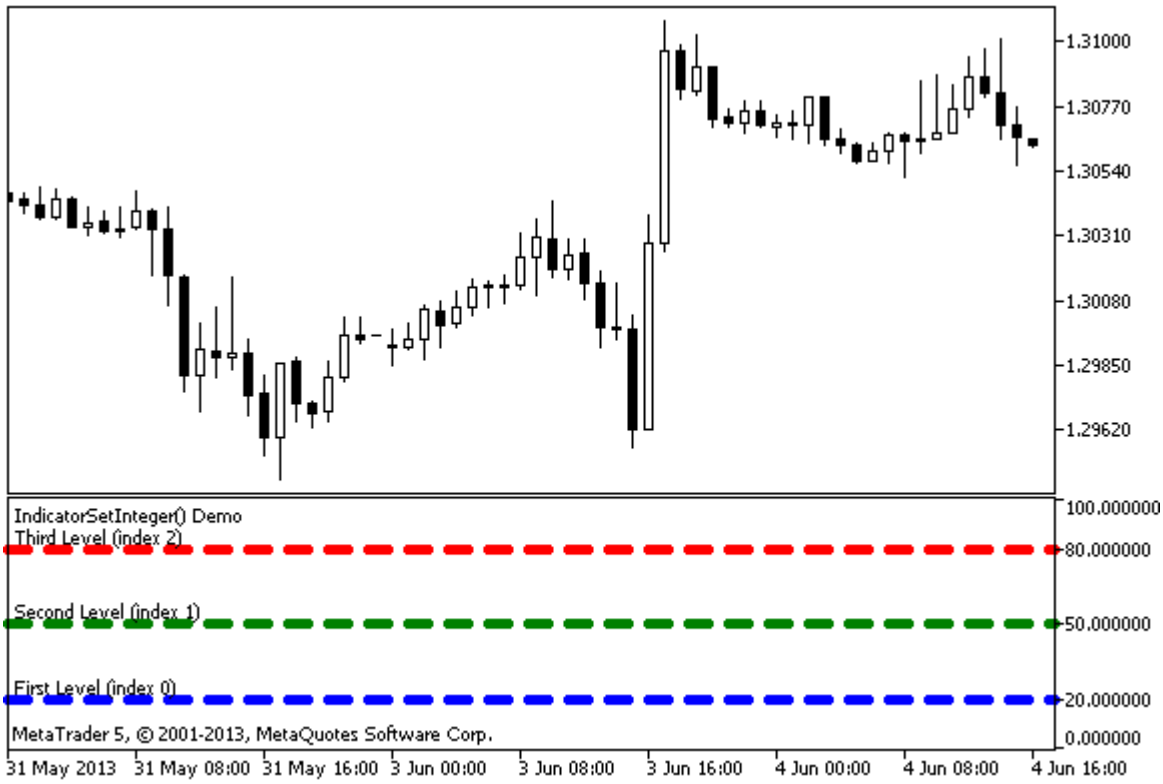
Numeração das propriedades (modificadores) começa a partir de 1 (um) quando usa a diretiva de #property, enquanto que a função usa numeração a partir de (zero). No caso do número de nível ser definido incorretamente, o visor do indicador pode ser diferente do pretendido.

Por exemplo, para definir a espessura da primeira linha horizontal usar o índice zero:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - índice 0 é usado para definir a espessura do primeiro nível.

Exemplo: indicador que define a cor, estilo e espessura do indicador de linhas horizontais.

© 2000-2024, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- visualiza três níveis horizontais na janela separada
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- definir a espessura dos níveis horizontais
#property indicator_levelwidth 5
//--- definir a cor dos níveis horizontais
#property indicator_levelcolor clrAliceBlue
//--- definir o estilo dos níveis horizontais
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- definir as descrições dos níveis horizontais
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primeiro Nível (índice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Segundo Nível (índice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terceiro Nível (índice 2)");
//--- define o mome abreviado para o indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- calcular ticks
   tick_counter++;
//--- e calcular as cores dos níveis horizontais, dependendo do contador de tick
   ChangeLevelColor(0,tick_counter,3,6,10); // três últimos parâmetros estão comutando
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- modificar o estilo dos níveis horizontais
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- obter a largura como o restante da divisão inteira por 5 ticks
   int width=tick_counter%5;
//--- iteração em todos os níveis horizontais e definir espessura
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Definir a cor da linha horizontal na janela separada do indicador|
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // número de linha horizontal
                      int tick_number,// dividendo, para obter o número do restante da
                      int f_trigger,  // primeiro divisor de comutação de cor
                      int s_trigger,  // segundo divisor de comutação de cor
                      int t_trigger)  // terceiro divisor de comutação de cor
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- índice de cor a partir do array colors[]
   int index=-1;
//--- calcular o número de cor a partir do array colors[] para pintar linha horizontal
   if(tick_number%f_trigger==0)
      index=0;   // se tick_number divide por f_trigger sem o restante
   if(tick_number%s_trigger==0)
      index=1;   // se tick_number divide por s_trigger sem o restante
```

© 2000-2024, MetaQuotes Ltd.

```
   if(tick_number%t_trigger==0)
      index=2;   // se tick_number divide por t_trigger sem o restante
//--- se a cor está definida, estabeleça-a
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| Define estilo de linha horizontal na janela do indicador separado|
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // número de linha horizontal
                      int tick_number// número para obter o resto da divisão
                      )
  {
//--- array para armazenar estilos
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- índice de estilo a partir do array styles[]
   int index=-1;
//--- calcular o número a partir do array styles[] para definir a linha horizontal
   if(tick_number%50==0)
      index=5;   // se tick_number divide por 50 sem o restante, então o estilo é STYI
   if(tick_number%40==0)
      index=4;   // ... style is STYLE_DASHDOT
   if(tick_number%30==0)
      index=3;   // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;   // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;   // ... STYLE_SOLID
//--- se o estilo está definido, estabeleça-o
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Também Veja

Propriedades de Indicador Personalizado, Propriedades (#property) de programa, Estilos de Desenho

© 2000-2024, MetaQuotes Ltd.

IndicatorSetString

A função define o valor da propriedade do indicador correspondente. Propriedade do indicador deve ser do tipo string. Existem duas variantes da função.

Chamar especificando o identificador da propriedade.

```
bool  IndicatorSetString(
   int     prop_id,           // identificador
   string  prop_value         // valor a ser definido
   );
```

Chamar especificando o identificador e o modificador da propriedade.

```
bool  IndicatorSetString(
   int     prop_id,           // identificador
   int     prop_modifier,     // modificador
   string  prop_value         // valor a ser definido
   )
```

Parâmetros

prop_id

[in] Identificador da propriedade do indicador. O valor pode ser um dos valores do enumerador ENUM_CUSTOMIND_PROPERTY_STRING.

prop_modifier

[in] Modificador da propriedade específica. Somente as propriedades de nível requerem um modificador.

prop_value

[in] Valor de propriedade.

Valor do Retorno

Em caso de execução bem sucedida, retorna true, do contrário - false.

Observação

Numeração das propriedades (modificadores) começa a partir de 1 (um) quando usa a diretiva de #property, enquanto que a função usa numeração a partir de (zero). No caso do número de nível ser definido incorretamente, o visor do indicador pode ser diferente do pretendido.

Por exemplo, na ordem para definir a descrição da primeira linha horizontal usar o índice zero:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "Primeiro Nível") - índice 0 é usado para definir a descrição de texto do primeiro nível.

Exemplo: indicador que define etiquetas de texto para as linhas horizontais do indicador.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- visualiza três níveis horizontais na janela separada
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- definir a cor dos níveis horizontais
#property indicator_levelcolor clrRed
//--- definir o estilo dos níveis horizontais
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- definir as descrições dos níveis horizontais
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"Primeiro Nível (índice 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Segundo Nível (índice 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Terceiro Nível (índice 2)");
//--- define o mome abreviado para o indicador
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
```

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
```

Também Veja

Propriedades de Indicador Personalizado, Propriedades (#property) de Programa

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetDouble

A função define o valor da propriedade correspondente da linha do indicador correspondente. A propriedade indicador deve ser do tipo double.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // plotando o índice do estilo
   int     prop_id,        // propriedade identificador
   double  prop_value      // valor para ser definido
   );
```

Parâmetros

plot_index

[in] Índice de plotagem gráfica

prop_id

[in] O valor pode ser um dos valores do enumerador ENUM_PLOT_PROPERTY_DOUBLE.

prop_value

[in] O valor da propriedade.

Valor do Retorno

Se com sucesso, retorna true, do contrário false.

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetInteger

A função define o valor da propriedade correspondente da linha do indicador correspondente. A propriedade do indicador deve ser do tipo int, char, bool ou color. Existem 2 variantes da função.

Chamar indicando o identificador da propriedade.

```
bool  PlotIndexSetInteger(
   int  plot_index,        // plotagem do índice do estilo
   int  prop_id,           // propriedade identificador
   int  prop_value         // valor a ser definido
   );
```

Chamar indicando o identificador e o modificador da propriedade.

```
bool  PlotIndexSetInteger(
   int  plot_index,        // plotagem do índice do estilo
   int  prop_id,           // propriedade identificador
   int  prop_modifier,     // propriedade modificador
   int  prop_value         // valor a ser definido
   )
```

Parâmetros

plot_index

[in] Índice de plotagem gráfica

prop_id

[in] O valor pode ser um dos valores do enumerador ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador da propriedade específica. Propriedades de índice de cor apenas requerem um modificador.

prop_value

[in] O valor da propriedade.

Valor do Retorno

Se com sucesso, retorna true, do contrário false.

Exemplo: um indicador que desenha uma linha de três cores. O esquema de cores muda a cada 5 ticks.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plotar ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- buffers do indicador
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- obter manipulador MA
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| Obter índice de cor                                              |
//+------------------------------------------------------------------+
```

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// primeiro índice
   if(j<200) return(1);// segundo índice
   return(2); // terceiro índice
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- primeiro cálculo ou número de barras foi alterado
   if(prev_calculated==0)
     {
      //--- copia valores da MA dentro do buffer do indicador ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// cópia falhou - lançar fora
      //--- agora definir a cor da linha para cada barra
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- copia valores da MA dentro do buffer do indicador ColorLineBuffer
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// contando ticks
      if(ticks>=5)//é hora de mudar o esquema de cores
        {
         ticks=0; // zerar o contador
         modified++; // contador de mudanças de cor
         if(modified>=3)modified=0;// zerar o contador
         ResetLastError();
         switch(modified)
           {
```

```
            case 0:// primeiro esquema de cor
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Esquema de cor "+modified);
               break;
            case 1:// segundo esquema de cor
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Esquema de cor "+modified);
               break;
            default:// terceiro esquema de cor
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Esquema de cor "+modified);
            }
         }
      else
         {
          //--- definir posição inicial
          limit=prev_calculated-1;
          //--- Agora vamos definir a cor da linha para cada barra
          for(int i=limit;i<rates_total;i++)
             ColorBuffer[i]=getIndexOfColor(i);
         }
      }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

PlotIndexSetString

A função define o valor da propriedade correspondente da linha do indicador correspondente. A propriedade indicador deve ser do tipo string.

```
bool  PlotIndexSetString(
   int     plot_index,     // plotando o índice do estilo
   int     prop_id,        // propriedade identificador
   string  prop_value      // valor para ser definido
   );
```

Parâmetros

plot_index

[in] Índice de representação gráfica

prop_id

[in] O valor pode ser um dos valores do enumerador ENUM_PLOT_PROPERTY_STRING.

prop_value

[in] O valor da propriedade.

Valor do Retorno

Se com sucesso, retorna true, do contrário false.

© 2000-2024, MetaQuotes Ltd.

PlotIndexGetInteger

A função define o valor da propriedade correspondente da linha do indicador correspondente. A propriedade do indicador deve ser do tipo int, color, bool ou char. Existem 2 variantes da função.

Chamar indicando o identificador da propriedade.

```
int  PlotIndexGetInteger(
   int  plot_index,        // plotagem do índice do estilo
   int  prop_id,           // propriedade identificador
   );
```

Chamar indicando o identificador e o modificador da propriedade.

```
int  PlotIndexGetInteger(
   int  plot_index,        // plotagem do índice
   int  prop_id,           // propriedade identificador
   int  prop_modifier      // propriedade modificador
   )
```

Parâmetros

plot_index

[in] Índice de plotagem gráfica

prop_id

[in] O valor pode ser um dos valores do enumerador ENUM_PLOT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador da propriedade específica. Propriedades de índice de cor apenas requerem um modificador.

Observação

Função é projetado para extrair as configurações do desenho da linha da indicador apropriada. A função trabalha em conjunto com a função PlotIndexSetInteger para copiar as propriedades de desenho de uma linha para outra.

Exemplo: um indicador que colore as velas dependendo do dia da semana. Cores para cada dia são criados de uma maneira programada.

© 2000-2024, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plotar ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- buffers do indicador
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- definir o número de cores em buffer de cor
```

© 2000-2024, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- definir cores para o buffer de cores
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- define acuracidade
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- definir a cor de cada vela
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

Funções de Objeto

Este é o grupo de funções destinadas ao trabalho com objetos gráficos relativos a qualquer gráfico especifico.

As funções que definem as propriedades de objetos gráficos, assim como as operações ObjectCreate() e ObjectMove() para criar e mover objetos ao longo do gráfico são realmente usados para enviar comandos para o gráfico. Se essas funções são executadas com êxito, o comando é incluído na fila comum dos eventos gráficos. As alterações visuais nas propriedades de objetos gráficos são implementadas ao manipular a fila dos eventos de gráfico.

Assim, não espere uma atualização visual imediata de objetos gráficos após chamar essas funções. Geralmente, os objetos gráficos no gráfico são atualizados automaticamente pelo terminal após os eventos de alteração - uma nova chegada de cotação, redimensionando a janela do gráfico, etc. Use a função ChartRedraw() para forçar a atualização dos objetos gráficos.

Função	Ação
ObjectCreate	Cria um objecto de tipo específico para um gráfico específico
ObjectName	Retorna o nome de um objeto do tipo correspondente para um gráfico específico (gráfico de sub-janela específica)
ObjectDelete	Remove o objeto com o nome específico a partir de um gráfico específico (a partir de gráfico de sub-janela específica)
ObjectsDeleteAll	Remove todos os objetos com o tipo específico a partir de um gráfico específico (a partir de gráfico de sub-janela específica)
ObjectFind	Pesquisa um objeto com o ID especificado pelo nome
ObjectGetTimeByValue	Retorna o valor de tempo ao valor do preço do objeto especifico
ObjectGetValueByTime	Retorna o valor do preço de um objeto ao período especificado
ObjectMove	Muda as coordenadas do ponto de ancoragem do objeto especificado
ObjectsTotal	Retorna o número de objetos do tipo especifico no gráfico especificado (gráfico de sub-janela específica)
ObjectGetDouble	Retorna o valor double da propriedade do objeto correspondente
ObjectGetInteger	Retorna o valor integer da propriedade do objeto correspondente
ObjectGetString	Retorna o valor string da propriedade do objeto correspondente
ObjectSetDouble	Define o valor da propriedade do objeto correspondente
ObjectSetInteger	Define o valor da propriedade do objeto correspondente
ObjectSetString	Define o valor da propriedade do objeto correspondente
TextSetFont	Define a fonte para exibir o texto usando métodos de desenho (Arial 20, usado por padrão)
TextOut	Transfere o texto para um array personalizado (buffer) projetado para a criação de recursos gráficos

© 2000-2024, MetaQuotes Ltd.

Função	Ação
TextGetSize	Retorna a largura e a altura da string nas configurações das fontes usadas no momento

Todos os objetos gráficos devem ter um único nome dentro de um gráfico, incluindo suas sub-janelas. Mudança de nome de um objeto gráfico gera dois eventos: evento de exclusão de um objeto com o antigo nome, e evento de criação de um objeto com um novo nome.

Após um objeto ser criado ou uma propriedade de objeto ser modificada, é recomendado chamar a função ChartRedraw(), a qual comanda o terminal do cliente, forçando o desenho de um gráfico (e todos os objetos visíveis dentro dele).

© 2000-2024, MetaQuotes Ltd.

ObjectCreate

A função cria um objeto com o nome especifico, o tipo e as coordenadas iniciais na sub-janela de gráfico especifico. Durante a criação até 30 coordenadas podem ser especificadas.

```
bool  ObjectCreate(
   long         chart_id,      // identificador gráfico
   string       name,          // nome objeto
   ENUM_OBJECT  type,          // tipo objeto
   sub_window   nwin,          // índice janela
   datetime     time1,         // tempo do primeiro ponto de ancoragem
   double       price1,        // preço do primeiro ponto de ancoragem
   ...
   datetime     timeN=0,       // tempo de N ponto de ancoragem
   double       priceN=0,      // preço de N ponto de ancoragem
   ...
   datetime     time30=0,      // tempo do trigésimo ponto de ancoragem
   double       price30=0      // preço do trigésimo ponto de ancoragem
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto. O nome deve ser único dentro de um gráfico, incluindo suas sub-janelas.

type

[in] Tipo do e objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT.

sub_window

[in] Número de sub-janela do gráfico. 0 (zero) significa a principal janela do gráfico. A sub-janela especifica deve existir, caso contrário, a função retorna false.

time1

[in] A coordenada do tempo da primeira ancoragem.

price1

[in] A coordenada do preço do primeiro ponto de ancoragem.

timeN=0

[in] Tempo de N ponto de ancoragem.

priceN=0

[in] A coordenada do preço de N ponto de ancoragem.

time30=0

[in] A coordenada de tempo de N ponto de ancoragem.

price30=0

[in] A coordenada do preço do trigésimo ponto de ancoragem.

© 2000-2024, MetaQuotes Ltd.

Valor do Retorno

Retorna true, se a colocação do comando na fila do gráfico especificado for bem-sucedida, caso contrário, false. Se o objeto já foi criado, é feita uma tentativa de alterar sua posição.

Observação

Ao chamar ObjectCreate(), sempre é usada uma chamada assíncrona, portanto a função retorna apenas o resultado da colocação do comando na fila do gráfico. Neste caso, true indica apenas que o comando está na fila com sucesso; e em si o resultado de sua execução não é conhecido.

Para verificação do resultado da execução, é possível utilizar a função ObjectFind() ou quaisquer funções que solicitam as propriedades do objeto, por exemplo, do tipo ObjectGetXXX. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos do gráfico e são esperados os resultados da execução (uma vez que são chamadas síncronas), isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

An object name should not exceed 63 characters.

A numeração das sub-janelas do gráfico (se houver sub-janelas com indicadores no gráfico) começa com 1. A janela principal do gráfico sempre tem índice 0.

O maior número de pontos de ancoragem (até 30) é implementado para uso futuro. Ao mesmo tempo, o limite de 30 possíveis pontos de ancoragem para os objetos gráficos é determinado pela limitação do número de parâmetros (não mais do que 64), que pode ser usado quando chama uma função.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

Há um certo número de pontos de ancoragem que devem ser especificados durante a criação de cada tipo de objeto:

ID	Description	Pontos de Ancoragem
OBJ_VLINE	Linha Vertical	Um ponto de ancoragem. Efetivamente apenas as coordenadas de tempo são usadas.
OBJ_HLINE	Linha Horizontal	Um ponto de ancoragem. Efetivamente apenas a coordenada de preço é usada.
OBJ_TREND	Linha de Tendência	Dois pontos para Ancoragem.
OBJ_TRENDBYANGLE	Linha de Tendência por Ângulo	Dois pontos para Ancoragem.
OBJ_CYCLES	Linhas de Ciclo	Dois pontos para Ancoragem.
OBJ_ARROWED_LINE	Linha de Seta	Dois pontos para Ancoragem.

© 2000-2024, MetaQuotes Ltd.

ID	Description	Pontos de Ancoragem
OBJ_CHANNEL	Canal Eqüidistante	Três pontos para Ancoragem.
OBJ_STDDEVCHANNEL	Canal de Desvio Padrão	Dois pontos para Ancoragem.
OBJ_REGRESSION	Canal de Regressão Linear	Dois pontos para Ancoragem.
OBJ_PITCHFORK	Forquilha de Andrews	Três pontos para Ancoragem.
OBJ_GANNLINE	Linha Gann	Dois pontos para Ancoragem.
OBJ_GANNFAN	Ventilador Gann	Dois pontos para Ancoragem.
OBJ_GANNGRID	Grade Gann	Dois pontos para Ancoragem.
OBJ_FIBO	Retração de Fibonacci	Dois pontos para Ancoragem.
OBJ_FIBOTIMES	Fusos Horários de Fibonacci	Dois pontos para Ancoragem.
OBJ_FIBOFAN	Fibonacci Fan	Dois pontos para Ancoragem.
OBJ_FIBOARC	Arcos de Fibonacci	Dois pontos para Ancoragem.
OBJ_FIBOCHANNEL	Canal de Fibonacci	Três pontos para Ancoragem.
OBJ_EXPANSION	Expansão de Fibonacci	Três pontos para Ancoragem.
OBJ_ELLIOTWAVE5	Onda Motriz de Elliott	Dois pontos para Ancoragem.
OBJ_ELLIOTWAVE3	Onde de Correção de Elliott	Três pontos para Ancoragem.
OBJ_RECTANGLE	Rectangle	Dois pontos para Ancoragem.
OBJ_TRIANGLE	Triangle	Três pontos para Ancoragem.
OBJ_ELLIPSE	Ellipse	Três pontos para Ancoragem.
OBJ_ARROW_THUMB_UP	Polegares para cima	Um ponto de ancoragem.
OBJ_ARROW_THUMB_DOWN	Polegar para Baixo	Um ponto de ancoragem.
OBJ_ARROW_UP	Seta para cima	Um ponto de ancoragem.
OBJ_ARROW_DOWN	Seta para baixo	Um ponto de ancoragem.
OBJ_ARROW_STOP	Sinal Stop	Um ponto de ancoragem.
OBJ_ARROW_CHECK	Conferido	Um ponto de ancoragem.
OBJ_ARROW_LEFT_PRICE	Etiqueta de Preço à Esquerda	Um ponto de ancoragem.
OBJ_ARROW_RIGHT_PRICE	Rótulo de Preço Direito	Um ponto de ancoragem.
OBJ_ARROW_BUY	Comprar	Um ponto de ancoragem.
OBJ_ARROW_SELL	Vender	Um ponto de ancoragem.
OBJ_ARROW	Seta	Um ponto de ancoragem.
OBJ_TEXT	Text	Um ponto de ancoragem.

© 2000-2024, MetaQuotes Ltd.

ID	Description	Pontos de Ancoragem
OBJ_LABEL	Rótulo	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_BUTTON	Botão	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_CHART	Gráfico	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_BITMAP	Bitmap	Um ponto de ancoragem.
OBJ_BITMAP_LABEL	Rótulo Bitmap	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_EDIT	Edit	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.
OBJ_EVENT	O objeto "Evento" correspondendo a um evento no calendário econômico	Um ponto de ancoragem. Efetivamente apenas as coordenadas de tempo são usadas.
OBJ_RECTANGLE_LABEL	O objeto "Rótulo de Retângulo" para criação e design de interface gráfico customizada.	Posição será definida usando as propriedades OBJPROP_XDISTANCE e OBJPROP_YDISTANCE.

© 2000-2024, MetaQuotes Ltd.

ObjectName

A função retorna o nome do objeto correspondente no gráfico determinado, na sub-janela específica, de tipo específico.

```
string  ObjectName(
   long  chart_id,            // identificador gráfico
   int   pos,                 // número da lista de objetos
   int   sub_window=-1,       // índice janela
   int   type=-1              // tipo objeto
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

pos

[in] Número ordinal do objeto de acordo com o filtro específico pelo número e tipo de sub-janela.

sub_window=-1

[in] Número de sub-janela do gráfico. 0 significa janela do gráfico principal, -1 significa todas sub-janelas do gráfico, incluindo a janela principal.

type=-1

[in] Tipo de objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT. -1 significa todos os tipos.

Valor do Retorno

Nome do objeto é retornado em caso de sucesso.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

© 2000-2024, MetaQuotes Ltd.

ObjectDelete

A função remove o objeto com nome específico a partir de um gráfico especificado.

```
bool  ObjectDelete(
   long    chart_id,      // identificador gráfico
   string  name           // nome objeto
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto para ser deletado.

Valor do Retorno

Retorna true, se a colocação do comando na fila do gráfico especificado for bem-sucedida, caso contrário, false.

Observação

Ao chamar ObjectDelete(), sempre é usada uma chamada assíncrona, portanto a função retorna apenas o resultado da colocação do comando na fila do gráfico. Neste caso, true indica apenas que o comando está na fila com sucesso; e em si o resultado de sua execução não é conhecido.

Para verificação do resultado da execução, é possível utilizar a função ObjectFind() ou quaisquer funções que solicitam as propriedades do objeto, por exemplo, do tipo ObjectGetXXX. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos do gráfico e são esperados os resultados da execução (uma vez que são chamadas síncronas), isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

© 2000-2024, MetaQuotes Ltd.

ObjectsDeleteAll

Remove todos os objetos a partir de um gráfico específico, específica sub-janela do gráfico, do tipo especificado.

```
int  ObjectsDeleteAll(
   long  chart_id,           // identificador gráfico
   int   sub_window=-1,      // índice janela
   int   type=-1             // tipo objeto
   );
```

Remove todos os objetos do tipo especificado usando prefixo em nomes de objeto.

```
int  ObjectsDeleteAll(
   long           chart_id,   // ID do gráfico
   const string     prefix,   // prefixo no nome do objeto
   int        sub_window=-1,  // índice da janela
   int       object_type=-1   // tipo do objeto
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

prefix

[in] Prefixo no nome dos objetos. Todos os objetos cujos nomes começam com este conjunto de caracteres serão removidos do gráfico. Você pode especificar prefixo como 'name' ou 'name*' - ambas as variantes funcionarão da mesma forma. Se uma string vazia for especificada como o prefixo, os objetos com todos os nomes possíveis serão removidos.

sub_window=-1

[in] Número de sub-janelas do gráfico. 0 significa janela do gráfico principal, -1 significa todas sub-janelas do gráfico, incluindo a janela principal.

type=-1

[in] Tipo de objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT. -1 significa todos os tipos.

Valor do Retorno

Retorna o número de objetos excluídos. Para ler mais sobre o erro, chamar GetLastError().

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

© 2000-2024, MetaQuotes Ltd.

ObjectFind

A função pesquisa por um objeto com o nome específico no gráfico com o ID especifico.

```
int  ObjectFind(
   long    chart_id,      // identificador gráfico
   string  name           // nome objeto
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] O nome do objeto pesquisado.

Valor do Retorno

Se for bem sucedido, a função retorna o número da sub-janela (0 significa a janela principal do gráfico), no qual o objeto foi encontrado. Se o objeto não for encontrado, a função retorna um número negativo. Para ler mais sobre o erro, chamar GetLastError().

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

© 2000-2024, MetaQuotes Ltd.

ObjectGetTimeByValue

A função retorna o valor de tempo para o valor do preço específico de um objeto específico.

```
datetime  ObjectGetTimeByValue(
   long    chart_id,      // identificador gráfico
   string  name,          // nome objeto
   double  value,         // Preço
   int     line_id        // Número linha
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

value

[in] Valor do preço.

line_id

[in] Identificador de linha.

Valor do Retorno

O valor de tempo para o valor de determinado preço de um objeto específico.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Um objeto pode ter vários valores em uma coordenada de preço, portanto é necessário especificar o número da linha. Esta função aplica-se apenas aos seguintes objetos:

- Linha de Tendência (OBJ_TREND)
- Linha de tendência pelo ângulo (OBJ_TRENDBYANGLE)
- Linha de Gann(OBJ_GANNLINE)
- Canal eqüidistante (OBJ_CHANNEL) - 2 linhas
- Canal de regressão Linear (OBJ_REGRESSION) - 3 linhas
- Canal de desvio padrão (OBJ_STDDEVCHANNEL) - 3 linhas
- Linha com setas (OBJ_ARROWED_LINE)

Veja Também

Tipos de Objeto

© 2000-2024, MetaQuotes Ltd.

ObjectGetValueByTime

A função retorna o valor do preço para o valor de tempo indicado de um objeto específico.

```
double  ObjectGetValueByTime(
   long      chart_id,      // identificador gráfico
   string    name,          // nome objeto
   datetime  time,          // Tempo
   int       line_id        // Número Linha
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

time

[in] Valor de tempo.

line_id

[in] ID de Linha.

Valor do Retorno

O valor do preço para o valor de tempo indicado de um objeto específico.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Um objeto pode ter vários valores em uma coordenada de preço, portanto é necessário especificar o número da linha. Esta função aplica-se apenas aos seguintes objetos:

- Linha de Tendência (OBJ_TREND)
- Linha de tendência pelo ângulo (OBJ_TRENDBYANGLE)
- Linha de Gann(OBJ_GANNLINE)
- Canal eqüidistante (OBJ_CHANNEL) - 2 linhas
- Canal de regressão Linear (OBJ_REGRESSION) - 3 linhas
- Canal de desvio padrão (OBJ_STDDEVCHANNEL) - 3 linhas
- Linha com setas (OBJ_ARROWED_LINE)

Veja Também

Tipos de Objeto

© 2000-2024, MetaQuotes Ltd.

ObjectMove

A função altera as coordenadas do ponto de ancoragem do objeto específico.

```
bool  ObjectMove(
   long      chart_id,         // identificador gráfico
   string    name,             // nome objeto
   int       point_index,      // número ponto ancoragem
   datetime  time,             // Tempo
   double    price             // Preço
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

point_index

[in] Índice do ponto de ancoragem. O número de pontos de ancoragem depende do tipo de objeto.

time

[in] Coordenada de tempo do ponto de ancoragem selecionado.

price

[in] Coordenada de preço do ponto de ancoragem selecionado.

Valor do Retorno

Retorna true, se a colocação do comando na fila do gráfico especificado for bem-sucedida, caso contrário, false.

Observação

Ao chamar ObjectMove(), sempre é usada uma chamada assíncrona, portanto a função retorna apenas o resultado da colocação do comando na fila do gráfico. Neste caso, true indica apenas que o comando está na fila com sucesso; e em si o resultado de sua execução não é conhecido.

Para verificar o resultado da execução, pode-se usar a função que solicita as propriedades do objeto, por exemplo, do tipo ObjectGetXXX. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos do gráfico e são esperados os resultados da execução (uma vez que são chamadas síncronas), isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

© 2000-2024, MetaQuotes Ltd.

ObjectsTotal

A função retorna o número de objetos na tabela determinada, sub-janela especifica, do tipo especificado.

```
int  ObjectsTotal(
   long  chart_id,           // identificador gráfico
   int   sub_window=-1,      // índice janela
   int   type=-1             // tipo objeto
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

sub_window=-1

[in] Número de sub-janela do gráfico. 0 significa janela do gráfico principal, -1 significa todas sub-janelas do gráfico, incluindo a janela principal.

type=-1

[in] Tipo de objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT. -1 significa todos os tipos.

Valor do Retorno

O número de objetos.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

© 2000-2024, MetaQuotes Ltd.

ObjectSetDouble

A função define o valor da propriedade do objeto correspondente. A propriedade objeto deve ser do tipo double. Existem 2 variantes da função.

Define valor da propriedade, sem modificador

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propriedade
   double                        prop_value         // valor
   );
```

Define um valor de propriedade que indica o modificador

```
bool  ObjectSetDouble(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // propriedade
   int                           prop_modifier,     // modificador
   double                        prop_value         // valor
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador da propriedade específica. Ela indica o número do nível em ferramentas Fibonacci e no objeto gráfico Tridente de Andrew. A numeração dos níveis começa a partir do zero.

prop_value

[in] O valor da propriedade.

Valor do Retorno

A função retorna true somente se o comando para alterar as propriedades de um objeto gráfico foi enviado a um gráfico com sucesso. Do contrário ele retorna false. Para ler mais sobre o erro, chamar GetLastError().

Observação

© 2000-2024, MetaQuotes Ltd.

A função utiliza uma chamada assíncrona, isso significa que a função espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado e retorna o controle imediatamente.

Para verificar o resultado da execução, pode-se usar a função que solicita a propriedade especificada do objeto. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos de outro gráfico e são esperados os resultados da execução, isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Exemplo de criação de um objeto Fibonacci e adicionando um novo nível nele

```
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- arrays auxiliares
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- Copiar os preços de abertura - 100 barras mais recentes são suficientes
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("Falha ao copiar os valores da série de preços de Alta");
      return;
     }
//--- Copiar os preço de fechamento - mais de 100 barras são suficientes
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Falha ao copiar os valores da série de preços de Baixa");
      return;
     }
//--- Copiar os tempos de abertura para as últimas 100 barras
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Falha ao copiar os valores da série de preços de tempo");
      return;
     }
//--- Organizar o acesso aos dados copiados como para timeseries - retrocesso
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- Coordenadas do primeiro ponto de ancoragem do objeto Fibo
   price1=high[70];
   time1=time[70];
//--- Coordenadas do segundo ponto de ancoragem do objeto Fibo
   price2=low[50];
```

© 2000-2024, MetaQuotes Ltd.

```
   time2=time[50];

//--- Tempo para criar o objeto Fibo
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
   if(created) // Se o objeto for criado com sucesso
     {
      //--- definir a cor dos níveis de Fibo
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- a propósito, quanto níveis Fibo temos?
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---saída para Jornal => número de nível: valor level_desription
      for(int i=0;i<levels;i++)
        {
         Print(i,": ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- Tentar aumentar o número de níveis por unidade
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // falha ao alterar o número de níveis
        {
         Print("Falhou ao alterar o número de níveis da Fibo, erro ",GetLastError());
        }
      //--- apenas informa
      Print("Níveis de Fibo depois = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- definir um valor para um nível recém-criado
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // conseguiu definir um valor para o nível
        {
         Print("Definido com sucesso mais um nível Fibo");
         //--- Também não se esqueça de definir o nível da descrição
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- Obter o valor real do número de níveis no objeto Fibo
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("Níveis de Fibo depois de adicionar = ",levels);
         //--- mais uma vez a saída de todos os níveis - só para ter certeza
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }
        }
      else // Falha se você tentar aumentar o número de níveis no objeto Fibo
        {
         Print("Falhou ao definir mais um nível Fibo. Error ",GetLastError());
        }
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Veja Também

Tipos de Objetos, Propriedades de objeto

© 2000-2024, MetaQuotes Ltd.

ObjectSetInteger

A função define o valor da propriedade do objeto correspondente. A propriedade do objeto deve ser do tipo datetime, int, color, bool ou char. Existem 2 variantes da função.

Define valor da propriedade, sem modificador

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificador Gráfico
   string                            name,              // nome objeto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propriedade
   long                              prop_value         // valor
   );
```

Define um valor de propriedade que indica o modificador

```
bool  ObjectSetInteger(
   long                              chart_id,          // identificador Gráfico
   string                            name,              // nome objeto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // propriedade
   int                               prop_modifier,     // modificador
   long                              prop_value         // valor
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador da propriedade específica. Indica o número do nível em ferramentas Fibonacci e no objeto gráfico do tipo Tridente de Andrew. A numeração dos níveis começa a partir do zero.

prop_value

[in] O valor da propriedade.

Valor do Retorno

A função retorna true somente se o comando para alterar as propriedades de um objeto gráfico foi enviado a um gráfico com sucesso. Do contrário ele retorna false. Para ler mais sobre o erro, chamar GetLastError().

Observação

A função utiliza uma chamada assíncrona, isso significa que a função espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado e retorna o controle imediatamente.

© 2000-2024, MetaQuotes Ltd.

Para verificar o resultado da execução, pode-se usar a função que solicita a propriedade especificada do objeto. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos de outro gráfico e são esperados os resultados da execução, isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Um exemplo de como criar uma tabela de cores da Web

```
//+------------------------------------------------------------------+
//|                                              Tabela de Cores da Web|
//|                          Copyright 2011, MetaQuotes Software Corp |
//|                                        https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140       // largura de uma edição de objeto
#define Y_SIZE 33        // altura de uma edição de objeto
//+------------------------------------------------------------------+
//| Array de cores da Web                                            |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrO
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,clr
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| Criando e inicializando uma edição de objeto                     |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- gerar um nome para uma nova edição de objeto
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- criar uma nova edição de objeto
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Não é possível criar: '",name,"'");
```

```
      return;
     }
//--- definir coordenadas, largura e altura
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//--- definir a cor do texto
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- definir a cor de fundo
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- definir o texto
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criar tabela de cores 7x20 para edição de objetos
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Veja Também

Tipos de Objetos, Propriedades de objeto

© 2000-2024, MetaQuotes Ltd.

ObjectSetString

A função define o valor da propriedade do objeto correspondente. A propriedade do objeto deve ser do tipo string. Existem 2 variantes da função.

Define valor da propriedade, sem modificador

```
bool  ObjectSetString(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propriedade
   string                        prop_value         // valor
   );
```

Define um valor de propriedade que indica o modificador

```
bool  ObjectSetString(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // propriedade
   int                           prop_modifier,     // modificador
   string                        prop_value         // valor
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificador da propriedade específica. Indica o número do nível em ferramentas Fibonacci e no objeto gráfico do tipo Tridente de Andrew. A numeração dos níveis começa a partir do zero.

prop_value

[in] O valor da propriedade.

Valor do Retorno

A função retorna true somente se o comando para alterar as propriedades de um objeto gráfico foi enviado a um gráfico com sucesso. Do contrário ele retorna false. Para ler mais sobre o erro, chamar GetLastError().

Observação

A função utiliza uma chamada assíncrona, isso significa que a função espera a execução do comando que foi colocado com sucesso na fila do gráfico especificado e retorna o controle imediatamente.

© 2000-2024, MetaQuotes Ltd.

Para verificar o resultado da execução, pode-se usar a função que solicita a propriedade especificada do objeto. Mas, ao fazer isto, deve-se ter em mente que estas funções são colocadas na fila de espera dos comandos de outro gráfico e são esperados os resultados da execução, isso quer dizer que não podem consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

© 2000-2024, MetaQuotes Ltd.

ObjectGetDouble

A função retorna o valor da propriedade do objeto correspondente. A propriedade objeto deve ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  ObjectGetDouble(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificador propriedade
   int                           prop_modifier=0    // modificador propriedade, se i
   );
```

2. Retorna true ou false, dependendo do sucesso da função. Se bem sucedido, o valor da propriedade é colocado para receber uma variável passada como referência pelo último parâmetro.

```
bool  ObjectGetDouble(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // identificador propriedade
   int                           prop_modifier,     // modificador propriedade
   double&                       double_var         // aqui nós aceitamos o valor da
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_DOUBLE.

prop_modifier

[in] Modificador da propriedade específica. Para a primeira variante, o modificador tem valor padrão igual a 0. A maioria das propriedades não requerem um modificador. Indica o número do nível em ferramentas Fibonacci e no objeto gráfico do tipo Tridente de Andrew. A numeração dos níveis começa a partir do zero.

double_var

[out] Variável do tipo double que recebeu o valor do propriedade solicitada.

Valor do Retorno

Valor do tipo double para a primeira variante chamada.

Para a segunda variante, se essa propriedade é mantida e o valor foi colocado na variável double_var, a função retorna true, caso contrário, retorna false. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

© 2000-2024, MetaQuotes Ltd.

ObjectGetInteger

A função retorna o valor da propriedade do objeto correspondente. A propriedade do objeto deve ser do tipo datetime, int, color, bool ou char. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  ObjectGetInteger(
   long                              chart_id,          // identificador Gráfico
   string                            name,              // nome objeto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // identificador propriedade
   int                               prop_modifier=0    // modificador de propriedade,
   );
```

2. Retorna true ou false, dependendo do sucesso da função. Se bem sucedido, o valor da propriedade é colocado para receber uma variável passada como referência pelo último parâmetro.

```
bool  ObjectGetInteger(
   long                              chart_id,          // identificador Gráfico
   string                            name,              // nome objeto
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // identificador propriedade
   int                               prop_modifier,     // modificador propriedade
   long&                             long_var           // aqui nós aceitamos o valor d
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_INTEGER.

prop_modifier

[in] Modificador da propriedade específica. Para a primeira variante, o modificador tem valor padrão igual a 0. A maioria das propriedades não requerem um modificador. Indica o número do nível em ferramentas Fibonacci e no objeto gráfico do tipo Tridente de Andrew. A numeração dos níveis começa a partir do zero.

long_var

[out] Variável do tipo long que recebe o valor da propriedade requisitada.

Valor do Retorno

O valor long para a primeira chamada da variante.

A função retorna true para a segunda variante, se essa propriedade for mantida e o valor foi colocado na variável long_var, caso contrário, retorna false. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

© 2000-2024, MetaQuotes Ltd.

ObjectGetString

A função retorna o valor da propriedade do objeto correspondente. A propriedade do objeto deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  ObjectGetString(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificador propriedade
   int                           prop_modifier=0    // modificador de propriedade, s
   );
```

2. Retorna true ou false, dependendo do sucesso da função. Se bem sucedido, o valor da propriedade é colocado para receber uma variável passada como referência pelo último parâmetro.

```
bool  ObjectGetString(
   long                          chart_id,          // identificador Gráfico
   string                        name,              // nome objeto
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // identificador propriedade
   int                           prop_modifier,     // modificador propriedade
   string&                       string_var         // aqui nós aceitamos o valor da
   );
```

Parâmetros

chart_id

[in] Identificador do gráfico. Significa o gráfico atual.

nome

[in] Nome do objeto.

prop_id

[in] ID da propriedade do objeto. O valor pode ser um dos valores do enumerador ENUM_OBJECT_PROPERTY_STRING.

prop_modifier

[in] Modificador da propriedade específica. Para a primeira variante, o modificador tem valor padrão igual a 0. A maioria das propriedades não requerem um modificador. Indica o número do nível em ferramentas Fibonacci e no objeto gráfico do tipo Tridente de Andrew. A numeração dos níveis começa a partir do zero.

string_var

[out] Variável do tipo string que recebe o valor das propriedades requisitadas.

Valor do Retorno

Valor string para a primeira versão da chamada.

Retorna true para a segunda versão da chamada, se essa propriedade for mantida e o valor foi colocado na variável string_var, caso contrário, retorna false. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

Observação

A função utiliza uma chamada síncrona, isso significa que a função espera a execução de todos os comandos que foram colocados na fila do gráfico antes de sua chamada, e por isso este recurso pode consumir muito tempo. Deve ter isso em mente, se você estiver trabalhando com um grande número de objetos no gráfico.

Quando um objeto é renomeado, dois eventos são formados simultaneamente. Estes eventos podem ser manipulados em um Expert Advisor ou o indicador pela função OnChartEvent():

- um evento de exclusão de um objeto com o nome antigo;
- um evento de criação de um objeto com um novo nome.

© 2000-2024, MetaQuotes Ltd.

TextSetFont

A função define a fonte para exibir o texto usando métodos de desenho e retorna o resultado dessa operação. Fonte Arial com o tamanho -120 (12 pt) é usado por padrão.

```
bool  TextSetFont(
   const string  name,            // nome da fonte ou caminho para o arquivo da fonte
   int           size,            // tamanho da fonte
   uint          flags,           // combinação de sinalizadores
   int           orientation=0    // ângulo de inclinação do texto
   );
```

Parâmetros

name

[in] Nome da fonte no sistema ou o nome do recurso contendo a fonte ou o caminho ao arquivo da fonte no disco.

size

[in] O tamanho da fonte, que pode ser definida usando valores positivos e negativos. No caso de valores negativos, o valor é definido em décimos de um ponto e o tamanho do texto depende das configurações do sistema operacional ("escala padrão" ou "grande escala"). No caso de valores negativos, o valor é definido em décimos de um ponto e o tamanho do texto depende das configurações do sistema operacional ("escala padrão" ou "grande escala"). Veja a Nota abaixo para mais informações sobre as diferenças entre os modos.

flags

[in] Combinação de sinalizadores descrevendo o estilo da fonte.

orientation

[in] Inclinação horizontal do texto em relação ao eixo X, a unidade de medida é 0,1 grau. Isso significa que *orientação=450* está para inclinação igual a 45 graus.

Valor de retorno

Retorna verdadeiro se a fonte atual é instalado com êxito, caso contrário é falso. Possibilidade de erros de códigos:

- ERR_INVALID_PARAMETER(4003) - *name* apresenta NULL ou"" (string vazia),
- ERR_INTERNAL_ERROR(4001) - erro de sistema operacional (por exemplo, uma tentativa de criar uma fonte inexistente).

Observação

Se "::" é usado em nome da fonte, a fonte é baixada do recurso EX5. Se *name* nome da fonte é especificado com uma extensão, a fonte é baixada do arquivo, se o caminho começa com "\" ou "/", o arquivo é procurado em relação ao diretório MQL5. Caso contrário, ele é procurado em relação ao caminho do arquivo EX5 que chamou a função TextSetFont().

O tamanho da fonte é definido usando valores positivos ou negativos. Esse fato define a dependência do tamanho do texto a partir das configurações do sistema operacional (escala de tamanho).

- Se o tamanho é especificado por um número positivo, este tamanho é transformado em unidades de medidas físicas de um dispositivo (pixels) quando se muda de uma fonte lógica para uma

© 2000-2024, MetaQuotes Ltd.

física, e este tamanho corresponde à altura dos símbolos glifos escolhidos entre as fontes disponíveis. Este caso não é recomendado quando os textos exibidos pela função TextOut() e os exibidos pelo OBJ_LABEL ("Etiquetas"), onde objetos gráfico estão a ser utilizados em conjunto no gráfico.

- Se o tamanho é determinado por um número negativo, este número deve ser definido em décimos de um ponto lógico (-350 é igual a 35 pontos lógicos) dividido em 10. Um valor obtido é então transformado em unidades de medidas físicas de um dispositivo (pixels) e corresponde ao valor absoluto da altura de um símbolo escolhido a partir das fontes disponíveis. Multiplicar o do tamanho de fonte determinada nas propriedades do objeto por -10 para tornar o tamanho de um texto na tela semelhante ao de um objeto OBJ_LABEL.

Os sinalizadores podem ser usados com a combinação de sinalizadores de estilo com um dos sinalizadores especificando a largura da fonte. Nome dos sinalizadores são mostrados abaixo.

Sinalizadores para especificar o estilo da fonte

Flag	Description
FONT_ITALIC	Itálico
FONT_UNDERLINE	Sublinhado
FONT_STRIKEOUT	Riscado

Sinalizadores para especificar a largura da fonte

Flag
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

Também Veja

© 2000-2024, MetaQuotes Ltd.

Recursos, ResourceCreate(), ResourceSave(), TextOut()

© 2000-2024, MetaQuotes Ltd.

TextOut

A função exibe um texto num array personalizado (buffer) e retorna o resultado dessa operação. O array é projetado para criar o recurso gráfico.

```
bool  TextOut(
   const string       text,          // texto exibido
   int                x,             // coordenada X
   int                y,             // coordenada Y
   uint               anchor,        // tipo de âncora
   uint               &data[],       // buffer de saída
   uint               width,         // largura de buffer em pixels
   uint               height,        // altura de buffer em pixels
   uint               color,         // cor do texto
   ENUM_COLOR_FORMAT  color_format   // formato de cor para a saída
   );
```

Parâmetros

text

[in] Texto exibido que será escrito para o buffer. Somente o texto de uma linha é exibido.

x

[in] Coordenada X do ponto de ancoragem do texto exibido.

y

[in] Coordenada Y do ponto de ancoragem do texto exibido.

anchor

[in] O valor de saída dos 9 métodos pré-definidos de localização do ponto de ancoragem do texto exibido. O valor é definido por uma combinação de dois sinalizadores - sinalizadores de alinhamento de texto horizontal e vertical. Nomes dos sinalizadores estão listados na Nota abaixo.

data[]

[in] Buffer, na qual o texto é exibido. O buffer é usado para criar o recurso gráfico.

width

[in] Largura do buffer em pixels.

height

[in] Altura do buffer em pixels.

color

[in] Cor do texto.

color_format

[in] Formato de cor é definida pelo valor do enumerador ENUM_COLOR_FORMAT.

Valor de retorno

Retorna true se bem sucedido, caso contrário false.

Observação

© 2000-2024, MetaQuotes Ltd.

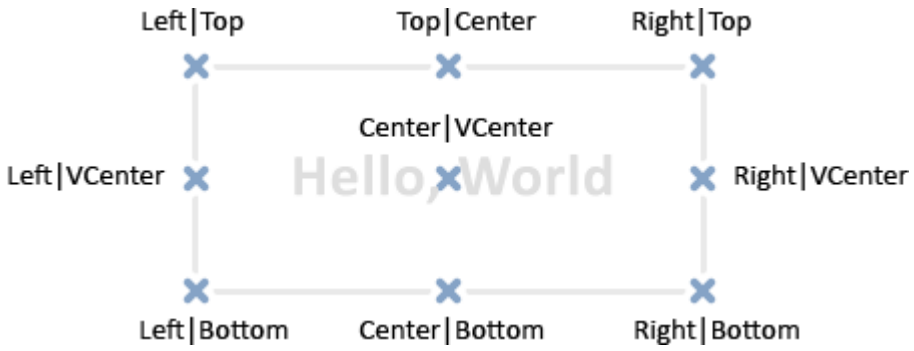
Ponto de ancoragem determinado por *anchor* é uma combinação de dois sinalizadores de alinhamento de texto horizontal e vertical. Sinalizador de alinhamento de texto horizontal:

- TA_LEFT - ponto de ancoragem no lado esquerdo da caixa delimitada
- TA_CENTER - ponto de ancoragem horizontal localiza-se no centro da caixa delimitada
- TA_RIGHT - ponto de ancoragem no lado direito da caixa delimitada

Sinalizador de alinhamento de texto vertical:

- TA_TOP - ponto de ancoragem no lado superior da caixa delimitada
- TA_VCENTER - ponto de ancoragem verticais está localizado no centro da caixa delimitada
- TA_BOTTOM - ponto de ancoragem no lado inferior da caixa delimitada

Combinações possíveis de sinalizadores e pontos de ancoragem determinados são mostrados na imagem.



Exemplo:

```
//--- largura e altura da tela (usado para o desenho)
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- exibir a janela de parâmetro antes de lançar o script
#property script_show_inputs
//--- permite definir o formato de cor
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- array de desenho (buffer)
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Programa Script da função start (iniciar)                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- criar o objeto OBJ_BITMAP_LABEL para desenhar
   ObjectCreate(0,"RELÓGIO",OBJ_BITMAP_LABEL,0,0,0);
//--- especificar o nome do recurso gráfico para escrever no objeto RELÓGIO
   ObjectSetString(0,"RELÓGIO",OBJPROP_BMPFILE,"::IMG");

//--- variáveis auxiliares
   double a;            // seta no canto
   uint   nm=2700;      // minutos no canto
   uint   nh=2700*12;   // horas no canto
   uint   w,h;          // variáveis para receber string de tamanhos de texto
```

© 2000-2024, MetaQuotes Ltd.

```
   int    x,y;          // variáveis para o cálculo das coordenadas atuais da string c

//--- girar ponteiros do relógio em um loop infinito, até que o script seja interromp
   while(!IsStopped())
     {
      //--- limpar o buffer do array do desenho do relógio
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- definir a fonte para desenhar dígitos para o rosto do relógio
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- desenhar a face do relógio
      for(int i=1;i<=12;i++)
        {
         //--- receber o tamanho da hora atual sobre a face do relógio
         TextGetSize(string(i),w,h);
         //--- calcular as coordenadas da hora atual sobre a face do relógio
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- saída da hora na face do relógio através do buffer ExtImg[]
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,c
        }
      //--- agora, especificar a fonte para desenhar o ponteiro dos minutos
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- receber o tamanho do ponteiro dos minutos
      TextGetSize("----->",w,h);
      //--- calcular as coordenadas do ponteiro dos minutos sobre a face do relógio
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- saída do ponteiro dos minutos na face do relógio através do buffer ExtImg
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //--- agora, definir a fonte para desenhar o ponteiro dos minutos
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- calcular as coordenadas do ponteiro das horas na face do relógio
      a=-(nh/12%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- saída do ponteiro das horas na face do relógio através do buffer ExtImg[]
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_form

      //--- atualizar o recurso gráfico
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- forçar a atualização do gráfico
      ChartRedraw();

      //--- aumentar os contadores da hora e minuto
      nm+=60;
      nh+=60;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- mantendo uma pequena pausa entre os quadros
      Sleep(10);
     }
//--- excluir o objeto RELÓGIO quando completar a operação do script
   ObjectDelete(0,"CLOCK");
//---
  }
```

Também Veja

Recursos, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2024, MetaQuotes Ltd.

TextGetSize

A função retorna a largura da linha e altura nas configurações das fonte atuais.

```
bool  TextGetSize(
   const string       text,          // string de texto
   uint&               width,        // largura do buffer em pixels
   uint&               height        // altura de buffer em pixels
   );
```

Parâmetros

text

[in] String, para o qual deve ser obtido o comprimento e a largura.

width

[out] Parâmetro de entrada para o recebimento da largura.

height

[out] Parâmetro de entrada para o recebimento da altura.

Valor de retorno

Retorna true se bem sucedido, caso contrário false. Possibilidade de erros de códigos:

- ERR_INTERNAL_ERROR(4001) - erro do sistema operacional.

Também Veja

Recursos, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2024, MetaQuotes Ltd.

Funções Indicadores Técnicos

Todas as funções, como iMA, iAC, iMACD, iIchimoku etc, tem uma cópia do indicador técnico correspondente criada no cache global do terminal do cliente. Se uma cópia do indicador com esses parâmetros já existe, a nova cópia não é criada, e o contador de referências para a cópia existente aumenta.

Essas funções retornam o manipulador da cópia apropriada do indicador. Além disso, usando este manipulador, você pode receber os dados calculados pelo indicador correspondente. Os dados do buffer correspondentes (indicadores técnicos contêm dados calculados nos seus buffers internos, que podem variar de 1 a 5, dependendo do indicador) podem ser copiados para um programa de MQL5 utilizando a função CopyBuffer().

Você não pode consultar os dados do indicador diretamente após ter sido criado, porque o cálculo de valores dos indicadores requer algum tempo, por isso é melhor criar o indicador manipulando em OnInit(). A Função iCustom() cria o indicador personalizado correspondente, e retorna seu manipulador no caso de ser criado com êxito. Indicadores personalizadas podem conter até 512 buffers de indicador, o conteúdo do qual pode também ser obtida através da função CopyBuffer(), usando o manipulador obtido.

Existe um método universal para a criação qualquer indicador técnico usando a função IndicatorCreate(). Esta função aceita os seguintes dados como parâmetros de entrada:

- Nome do símbolo;
- timeframe;
- tipo de indicador para criar;
- número de parâmetros de entrada do indicador;
- um array do tipo MqlParam contendo todos os parâmetros de entrada necessários.

A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(),onde o manipulador de indicador é transmitido.

Nota. Chamada repetida da função do indicador com os mesmos parâmetros dentro de um MQL5-programa não resulta num aumento múltiplo do contador de referência, o contador será aumentado apenas uma vez por 1. No entanto, é recomendado obter os manipuladores de indicadores em função OnInit() ou no construtor da classe, e ainda utilizar estes manipuladores em outras funções. O contador de referência diminui quando um MQL5-programa é desinicializado.

Todas as funções de indicador tem pelo menos dois parâmetros - o símbolo e o período. O NULL Valor do símbolo significa o símbolo corrente, o valor 0 (zero) do período significa o prazo corrente.

Função	Retorna o manipulador do indicador:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator

© 2000-2024, MetaQuotes Ltd.

Função	Retorna o manipulador do indicador:
iAMA	Adaptive Moving Average
iAO	Awesome Oscillator
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	Custom indicator
iDEMA	Double Exponential Moving Average
iDeMarker	DeMarker
iEnvelopes	Envelopes
iForce	Force Index
iFractals	Fractals
iFrAMA	Fractal Adaptive Moving Average
iGator	Gator Oscillator
iIchimoku	Ichimoku Kinko Hyo
iBWMFI	Market Facilitation Index by Bill Williams
iMomentum	Momentum
iMFI	Money Flow Index
iMA	Moving Average
iOsMA	Moving Average of Oscillator (MACD histogram)
iMACD	Moving Averages Convergence-Divergence
iOBV	On Balance Volume
iSAR	Parabolic Stop And Reverse System
iRSI	Relative Strength Index
iRVI	Relative Vigor Index
iStdDev	Standard Deviation
iStochastic	Stochastic Oscillator
iTEMA	Triple Exponential Moving Average

© 2000-2024, MetaQuotes Ltd.

Função	Retorna o manipulador do indicador:
iTriX	Triple Exponential Moving Averages Oscillator
iWPR	Williams' Percent Range
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

© 2000-2024, MetaQuotes Ltd.

iAC

A função cria o Accelerator Oscillator num cachê global do terminal do cliente e retorna seu manipulador. Tem apenas um buffer.

```
int  iAC(
   string           symbol,      // símbolo nome
   ENUM_TIMEFRAMES  period       // período
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores do enumerador ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iAC."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotagem do iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,                // usar iAC
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iAC;          // tipo da função
input string                symbol=" ";             // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // timeframe
//--- buffers do indicador
double iACBuffer[];
double iACColors[];
//--- variável para armazenamento do manipulador do indicador iAC
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- nós manteremos o número de valores no indicador Accelerator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
```

```
      {
       //--- mensagem sobre a falha e a saída do código de erro
       PrintFormat("Falha ao criar o manipulador do indicador iAC para o símbolo %s/%s,
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- o indicador é interrompido precocemente
       return(INIT_FAILED);
      }
//--- mostrar o símbolo/prazo do indicador Accelerator Oscillator, é calculado para
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iAC
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores em
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- Se o array iACBuffer é maior do que o número de valores no indicador iAC p
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays iACBuffer e iACColors com valores a partir do indicador Acce
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenchendo buffers do indicador a partir do indicador iAC       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // valores do buffer do indicator A
                          double &color_indexes[], // buffer de cor(para armazenar o i
                          int ind_handle,          // manipulador do indicador iAC
                          int amount               // número dos valores copiados
                          )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iACBuffer com valores a partir do buffer do indicad
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iAC, código de erro %d"
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- agora copiar o índice de cores
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar valores de cor a partir do indicador iAC, código de
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAD

A função retorna o manipulador do indicador Accumulation/Distribution. Tem apenas um buffer.

```
int  iAD(
   string               symbol,             // símbolo nome
   ENUM_TIMEFRAMES      period,             // período
   ENUM_APPLIED_VOLUME  applied_volume      // tipo de volume para o cálculo
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores do enumerador ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

applied_volume

[in] O volume utilizado. Can be any of ENUM_APPLIED_VOLUME values.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "dos buffers do indicador para o indicador técnico iAD."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,                // usar iAD
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iAD;          // tipo da função
input ENUM_APPLIED_VOLUME  volumes;                // volume usado
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffer do indicador
double         iADBuffer[];
//--- variável para armazenar o manipulador do indicador iAD
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Accumulation/Distribution
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iAD para o símbolo %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/prazo, o indicador de Accumulation/Distribution é calculado pa
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iAD
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores em
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se o array iADBuffer é maior do que o número de valores no indicador iAD p
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iADBuffer com valores do indicador Accumulation/Distribution
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Accumulation/Distribution
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenchendo buffers do indicador a partir do indicador iAD       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // valores do buffer do indicator da link
                         int ind_handle,     // manipulador do indicador iAD
                         int amount          // número dos valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iADBuffer com valores a partir do buffer do indicad
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iAD, código de erro %d"
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iADX

A função retorna o manipulador do indicador Average Directional Movement Index.

```
int  iADX(
   string           symbol,         // símbolo nome
   ENUM_TIMEFRAMES  period,         // período
   int              adx_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

adx_period

[in] Período para calcular o índice.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números do buffer são os seguintes: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iADX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "dos buffers do indicador para o indicador técnico iADX."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- plotar ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plotar DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // usar iADX
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iADX;         // tipo da função
input int                  adx_period=14;          // cálculo do período
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffers do indicador
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- variável para armazenar o manipulador do indicador iADX
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Average Directional Movement Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iADX para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/prazo, o indicador de Average Directional Movement Index é cal
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iADX
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores em
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iADXBuffer é maior do que o número de valores no indicador iADX
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Average Directional Movement Index
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Average Directional Movement Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenchendo buffers do indicador a partir do indicador iADX      |
//+------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // buffer do indicador da linha
                           double &DIplus_values[],   // buffer do indicador para DI+
                           double &DIminus_values[],  // buffer do indicador para DI-
                           int ind_handle,            // manipulador do indicador iADX
                           int amount                 // número dos valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iADXBuffer com valores a partir do buffer do indica
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iADX, código de erro %d
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array DI_plusBuffer com valores a partir do buffer do ind
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iADX, código de erro %d
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array DI_plusBuffer com valores a partir do buffer do ind
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iADX, código de erro %d
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iADXWilder

A função retorna o manipulador de Average Directional Movement Index por Welles Wilder.

```
int  iADXWilder(
   string           symbol,         // símbolo nome
   ENUM_TIMEFRAMES  period,         // período
   int              adx_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

adx_period

[in] Período para calcular o índice.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números do buffer são os seguintes: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   iADXWilder.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "dos buffers do indicador  para o indicador técnico iADXWilder."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
```

```
//--- plotar ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plotar DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // usar iADXWilder
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iADXWilder;   // tipo da função
input int                  adx_period=14;          // cálculo do período
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffers do indicador
double         ADXBuffer[];
double         DI_plusBuffer[];
double         DI_minusBuffer[];
//--- variável para armazenar o manipulador do indicador iADXWilder
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Average Directional Movement Index p
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iADXWilder para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/prazo, o indicador de Average Directional Movement Index por W
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
```

```
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iADXWilder
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores em
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iADXBuffer é maior do que o número de valores no indicador iADX
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Average Directional Movement Index po
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Average Directional Movement Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenchendo buffers do indicador a partir do indicador iADXWilder|
//+------------------------------------------------------------------+
```

```
bool FillArraysFromBuffers(double &adx_values[],      // buffer do indicador da linha
                           double &DIplus_values[],   // buffer do indicador para DI+
                           double &DIminus_values[],  // buffer do indicador para DI-
                           int ind_handle,            // manipulador do indicador iADX
                           int amount                 // número dos valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iADXBuffer com valores a partir do buffer do indica
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %d",G
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array DI_plusBuffer com valores a partir do buffer do ind
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %d",G
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array DI_plusBuffer com valores a partir do buffer do ind
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Failed to copy data from the iADXWilder indicator, error code %d",G
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAlligator

A função retorna o manipulador do indicador Alligator.

```
int  iAlligator(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 jaw_period,        // período para cálculo da mandíbula
   int                 jaw_shift,         // deslocamento horizontal da mandíbula
   int                 teeth_period,      // período para cálculo dos dentes
   int                 teeth_shift,       // deslocamento horizontal dos dentes
   int                 lips_period,       // período para o cálculo da boca
   int                 lips_shift,        // deslocamento horizontal da boca
   ENUM_MA_METHOD      ma_method,         // tipo de suavização
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

jaw_period

[in] Período médio para a linha azul (mandíbula do Alligator)

jaw_shift

[in] O deslocamento da linha azul em relação à tabela de preços.

teeth_period

[in] Período médio para a linha vermelha (dentes do Alligator).

teeth_shift

[in] O deslocamento da linha vermelha em relação para a tabela de preços.

lips_period

[in] Período médio para a linha verde (boca de Alligator).

lips_shift

[in] O deslocamento da linha verde em relação à tabela de preços.

ma_method

[in] O método da média. Pode ser qualquer um dos valores do ENUM_MA_METHOD.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

© 2000-2024, MetaQuotes Ltd.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números do buffer são os seguintes: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "dos buffers do indicador para o indicador técnico iAlligator."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares aos do padrão Alligato

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
```

```
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,        // usar iAlligator
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iAlligator;   // tipo de função
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
input int                  jaw_period=13;          // período da linha da mandíbula
input int                  jaw_shift=8;            // deslocamento da linha da mandíbu
input int                  teeth_period=8;         // período da linha dos dentes
input int                  teeth_shift=5;          // deslocamento da linha dos dentes
input int                  lips_period=5;          // período da linha da boca
input int                  lips_shift=3;           // deslocamento da linha da boca
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // método das linhas médias do All
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// tipo de preço utilizado para
//--- buffers do indicador
double         JawsBuffer[];
double         TeethBuffer[];
double         LipsBuffer[];
//--- variável para armazenar o manipulador do indicador
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores dentro do indicador Alligator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- definir o deslocamento de cada linha
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[8];
     //--- períodos e deslocamentos das linhas do Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//--- tipo de suavização
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- tipo de preço
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- criar manipulador
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iAlligator para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/prazo do indicador Alligator calculado para
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
```

© 2000-2024, MetaQuotes Ltd.

```
                              jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iAlligator
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores qu
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- Se o array JawsBuffer é maior do que o número dos valores do indicador iAl
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com os valores do indicador de Alligator
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2024, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores dentro do indicador Alligator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iAlligator  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // buffer do indicador para a linha
                           int j_shift,            // deslocamento da linha da mandíbu
                           double &teeth_buffer[], // buffer do indicador para a linha
                           int t_shift,            // deslocamento da linha dos dentes
                           double &lips_buffer[],  // buffer do indicador para a linha
                           int l_shift,            // deslocamento da linha da boca
                           int ind_handle,         // manipulador do indicator iAlliga
                           int amount              // número de valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array JawsBuffer com valores a partir do buffer do indica
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha para copiar dados a partir do indicador iAlligator, código de
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array TeethBuffer com valores a partir do buffer do indic
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha para copiar dados a partir do indicador iAlligator, código de
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array LipsBuffer com valores a partir do buffer do indica
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha para copiar dados a partir do indicador iAlligator, código de
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAMA

A função retorna o manipulador do indicador Adaptive Moving Average. Tem apenas um buffer.

```
int  iAMA(
   string              symbol,             // símbolo nome
   ENUM_TIMEFRAMES     period,             // período
   int                 ama_period,         // período médio para AMA
   int                 fast_ma_period,     // período rápido MA
   int                 slow_ma_period,     // período lento MA
   int                 ama_shift,          // deslocamento horizontal do indicador
   ENUM_APPLIED_PRICE  applied_price       // tipo do preço ou manipular
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ama_period

[in] O período de cálculo, em que o coeficiente de eficiência é calculado.

fast_ma_period

[in] Período rápido para o cálculo do coeficiente de suavização de um mercado de rápido.

slow_ma_period

[in] Período lento para o cálculo do coeficiente de suavização na ausência de tendência.

ama_shift

[in] Deslocamento do indicador relativo ao gráfico de preços.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iAMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são semelhantes aos do padrão AMA."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // use iAMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iAMA;          // tipo de função
input string               symbol=" ";              // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // timeframe
input int                  ama_period=15;           // período para cálculo
input int                  fast_ma_period=2;        // período da MA rápida
input int                  slow_ma_period=30;       // período da MA lenta
input int                  ama_shift=0;             // deslocamento horizontal
input ENUM_APPLIED_PRICE   applied_price;           // tipo de preço
//--- buffer do indicador
double         iAMABuffer[];
//--- variável para armazenar o manipulador do indicador iAMA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- mapeamento de buffers do indicador
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- tipo de preço
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o indicador iAMA para o símbolo %s/%s, código de er
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/prazo, o indicador Adaptive Moving Average é calculado para
```

© 2000-2024, MetaQuotes Ltd.

```
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iAMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se é o primeiro arranque do cálculo do indicador ou se o número de valores do in
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iAMABuffer é maior do que o número de valores do indicador iAMA
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com os valores do indicador Adaptive Moving Average
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2024, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Adaptive Moving Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher o buffer do indicador a partir do indicador iAMA       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // buffer do indicador da linha AMA
                         int a_shift,           // deslocamento da linha AMA
                         int ind_handle,        // manipulador do indicador iAMA
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- Preencher a partir do array iAMABuffer com os valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados a partir do indicador iAMA, código de erro %d
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iAO

A função retorna o manipulador do indicador Awesome Oscillator. Tem apenas um buffer.

```
int  iAO(
   string           symbol,      // símbolo nome
   ENUM_TIMEFRAMES  period       // período
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iAO."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar iAO
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
```

```
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,                // usar iAO
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iAO;          // tipo de função
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffers do indicador
double         iAOBuffer[];
double         iAOColors[];
//--- variável para armazenar o manipulador do indicador iAO
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Awesome Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
```

```
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iAO para o símbolo %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar o símbolo/período o indicador Awesome Oscillator é calculado para
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiado do indicator iAO
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o primeiro arranque do cálculo do indicador, ou se o número de valores no
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iAOBuffer é maior do que os valores no indicador iAO para símbo
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
```

```
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays iAOBuffer e iAOColors com valores a partir do indicador Awes
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher os buffers do indicador a partir do indicador iAO      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // buffer do indicador dos valores
                          double &color_indexes[], // buffer de cor(para armazenar o i
                          int ind_handle,          // manipulador do indicador iAO
                          int amount               // número dos valores copiados
                          )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iAOBuffer com valores a partir do buffer do indicad
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iAO, código de erro %d",GetLastE
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- agora copiar o índice de cores
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar valores de cor do indicador iAO, código de erro %d"
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iATR

A função retorna o manipulador do indicador Average True Range. Tem apenas um buffer.

```
int  iATR(
   string           symbol,        // símbolo nome
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] O valor do período médio para cálculo do indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// usar iATR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input int                  atr_period=14;          // cálculo do período
input Creation             type=Call_iATR;          // tipo da função
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffer do indicador
double         iATRBuffer[];
//--- variável para armazenamento do manipulador do indicador iAC
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número dos valores do indicador Average True Range indicator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iATR para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador True Range é calculado para
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iATR
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se é o princípio de cálculo do indicador ou se é o número de valores modificados
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se o array iATRBuffer é maior do que o número de valores no indicador iATR
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iATRBuffer com valores do indicador Average True Range
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Accelerator Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preenchendo os buffers do indicator a partir do indicador iATR   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicador para valores ATR va
                         int ind_handle,    // manipulador do indicador iATR
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher parte do array iATRBuffer com valores a partir do buffer do indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar a partir do indicador iATR , código de erro %d",Get
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBearsPower

A função retorna o manipulador do indicador Bears Power. Tem apenas um buffer.

```
int  iBearsPower(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] O valor do período médio para cálculo do indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iBearsPower."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iBearsPower
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // usar iBearsPower
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iBearsPower;  // tipo da função
input int                  ma_period=13;           // período da média móvel
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffer do indicador
double         iBearsPowerBuffer[];
//--- variável para armazenar o manipulador do indicador iBearsPower
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Bears Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- período da média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iBearsPower para o símbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar que o símbolo/período do indicador Bears Power é calculado para
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iBearsPower
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- se o array iBearsPowerBuffer é maior do que o número de valores no indicad
       //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
       //--- para o cálculo não mais do que uma barra é adicionada
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- preencher o array iBearsPowerBuffer com os valores do indicador Bears Power
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
    if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
    string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- exibir a mensagem de serviço no gráfico
    Comment(comm);
//--- memorizar o número de valores no indicador Bears Power
    bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iBearsPower |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicador para valores do ind
                         int ind_handle,    // manipulador do indicator iBearsPower
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
    ResetLastError();
//--- preencher uma parte do array iBearsPowerBuffer com valores do indicador buffer c
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- Se a cópia falhar, informe o código de erro
       PrintFormat("Falha ao copiar dados do indicador iBearsPower, código de erro %d",
       //--- parar com resultado zero - significa que indicador é considerado como não
       return(false);
      }
//--- está tudo bem
    return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBands

A função retorna o manipulador do indicador Bollinger Bands®.

```
int  iBands(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 bands_period,      // período para o cálculo da média da linha
   int                 bands_shift,       // deslocamento horizontal do indicador
   double              deviation,         // número de desvios padrão
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

bands_period

[in] O período médio da linha principal do indicador.

bands_shift

[in] O deslocamento do indicador em relação ao gráfico de preço.

deviation

[in] Desvio a partir da linha principal.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer são os seguintes: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBands.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "buffers do indicador para o indicador técnico iBands."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plotar linha superior
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar linha inferior
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plotar linha média
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,            // usar iBands
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iBands;          // tipo da função
input int                  bands_period=20;           // período da média móvel
input int                  bands_shift=0;             // deslocamento
input double               deviation=2.0;             // número de desvios padrão
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffers do indicador
double         UpperBuffer[];
double         LowerBuffer[];
```

```
double         MiddleBuffer[];
//--- variável para armazenar o manipulador do indicador iBands
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Bollinger Bands
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- definir o deslocamento de cada linha
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período da média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- número do desvio padrão
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iBands para o símbolo %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Bollinger Bands é calculado para
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iBands
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- se o tamanho buffers do indicador é maior do que o número de valores no i
       //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
       //--- para o cálculo não mais do que uma barra é adicionada
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- preencher o array com valores do indicador Bollinger Bands
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
    if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- formar a mensagem
    string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- exibir a mensagem de serviço no gráfico
    Comment(comm);
//--- memorizar o número de valores no indicador Bollinger Bands
    bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iBands      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // buffer do indicador da linha
                           double &upper_values[],    // buffer do indicador da borda
                           double &lower_values[],    // buffer do indicador da borda
                           int shift,                 // deslocamento
                           int ind_handle,            // manipulador do indicador iBan
                           int amount                 // número dos valores copiados
                           )
  {
//--- redefinir o código de erro
    ResetLastError();
//--- preencher uma parte do array MiddleBuffer com valores do buffer do indicador que
    if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
      {
       //--- Se a cópia falhar, informe o código de erro
       PrintFormat("Falha ao copiar dados do indicador iBands, código de erro %d",GetLa
       //--- parar com resultado zero - significa que indicador é considerado como não
       return(false);
      }
```

```
//--- preencher uma parte do array UpperBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iBands, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array LowerBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iBands, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBullsPower

A função retorna o manipulador do indicador Bulls Power. Tem apenas um buffer.

```
int  iBullsPower(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] O período médio para cálculo do indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iBullsPower."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iBullsPower
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // usar iBullsPower
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iBullsPower;  // tipo da função
input int                  ma_period=13;           // período da média móvel
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
//--- buffer do indicador
double         iBullsPowerBuffer[];
//--- variável para armazenar o manipulador do indicador iBullsPower
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Bulls Power
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- período da média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iBullsPower para o símbol
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostrar que o símbolo/período do indicador Bulls Power é calculado para
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iBullsPower
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
      {
       //--- se o array iBullsPowerBuffer é maior do que o número de valores no indicad
       //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
       //--- para o cálculo não mais do que uma barra é adicionada
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- preencher o array iBullsPowerBuffer com os valores do indicador Bulls Power
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
    if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
    string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- exibir a mensagem de serviço no gráfico
    Comment(comm);
//--- memorizar o número de valores no indicador Bulls Power
    bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iBullsPower |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicador para valores do ind
                         int ind_handle,    // manipulador do indicator iBullsPower
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
    ResetLastError();
//--- preencher uma parte do array iBullsPowerBuffer com valores do indicador buffer c
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
      {
       //--- Se a cópia falhar, informe o código de erro
       PrintFormat("Falha ao copiar dados do indicador iBullsPower, código de erro %d",
       //--- parar com resultado zero - significa que indicador é considerado como não
       return(false);
      }
//--- está tudo bem
    return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

iCCI

A função retorna o manipulador do indicador Commodity Channel Index. Tem apenas um buffer.

```
int  iCCI(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou de manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] O período médio para o cálculo de indicadores.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iCCI."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- plotar iCCI
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- níveis horizontais na janela de indicador
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // usar iCCI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iCCI;               // tipo da função
input int                  ma_period=14;                 // período da média móvel
input ENUM_APPLIED_PRICE   applied_price=PRICE_TYPICAL;  // tipo de preço
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iCCIBuffer[];
//--- variável para armazenar o manipulador do indicator iCCI
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Commodity Channel Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de preço
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iCCI para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador iCCI é calculado para
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- número de valores copiados a partir do indicador iCCI
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iCCIBuffer é maior do que o número de valores no indicador iCCI
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iCCIBuffer com valores do indicador Commodity Channel Index
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Commodity Channel Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iCCI        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicador de valores do Commo
                         int ind_handle,    // manipulador do indicador iCCI
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
```

```
   ResetLastError();
//--- preencher uma parte do array iCCIBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iCCI, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iChaikin

A função retorna o manipulador do indicador Chaikin Oscillator. Tem apenas um buffer.

```
int  iChaikin(
   string                symbol,             // símbolo nome
   ENUM_TIMEFRAMES       period,             // período
   int                   fast_ma_period,     // período rápido
   int                   slow_ma_period,     // período lento
   ENUM_MA_METHOD        ma_method,          // tipo suavizado
   ENUM_APPLIED_VOLUME   applied_volume      // tipo de volume
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

fast_ma_period

[in] Período médio rápido para cálculos.

slow_ma_period

[in] Período médio lento para cálculos.

ma_method

[in] Tipo suavizado. Pode ser uma das médias constantes de ENUM_MA_METHOD.

applied_volume

[in] O volume usado. Pode ser uma das constantes do ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iChaikin.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
```

```
#property description "de buffers do indicador para o indicador técnico iChaikin."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iChaikin
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // usar iChaikin
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iChaikin;           // tipo da função
input int                   fast_ma_period=3;             // período rápido da média mó
input int                   slow_ma_period=10;            // período lento da média móv
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // tipo de suavização
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // tipo de volume
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iChaikinBuffer[];
//--- variável para armazenar o manipulador do indicator iChaikin
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Chaikin Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
```

```
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período rápido da média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- período lento da média móvel
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- tipo de suavização
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de volume
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iChaikin para o símbolo %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Chaikin Oscillator é calculado para
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iChaikin
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iCCIBuffer é maior do que o número de valores no indicador iCha
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iChaikinBuffer com valores do indicador Chaikin Oscillator
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Chaikin Oscillator
```

© 2000-2024, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iChaikin    |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Cha
                         int ind_handle,    // manipulador do indicador iChaikin
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iChaikinBuffer com valores do buffer do indicador c
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iChaikin, código de erro %d",Get
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iCustom

A função retorna o manipulador de um indicador personalizado especifico.

```
int  iCustom(
   string           symbol,      // símbolo nome
   ENUM_TIMEFRAMES  period,      // período
   string           name         // folder/nome_do_indicador_personalizado
   ...                           // lista de parâmetros de entrada do indicador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

name

[in] O nome do indicador personalizado, com o caminho relativo ao diretório raiz de indicadores (MQL5\Indicators\). Se um indicador está localizado em um subdiretório, por exemplo, em MQL5/Indicadores/**Exemplos**, o seu nome deve ser especificado como: "**Examples**\\nome_do_indicador" (é necessário utilizar uma barra dupla, em vez de uma única barra como um separador).

...

[in] entrada-parâmetros de um indicador personalizado, separados por vírgulas. Tipo e ordem dos parâmetros devem coincidir. Se não há parâmetros específicos, então valores padrões serão usados.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

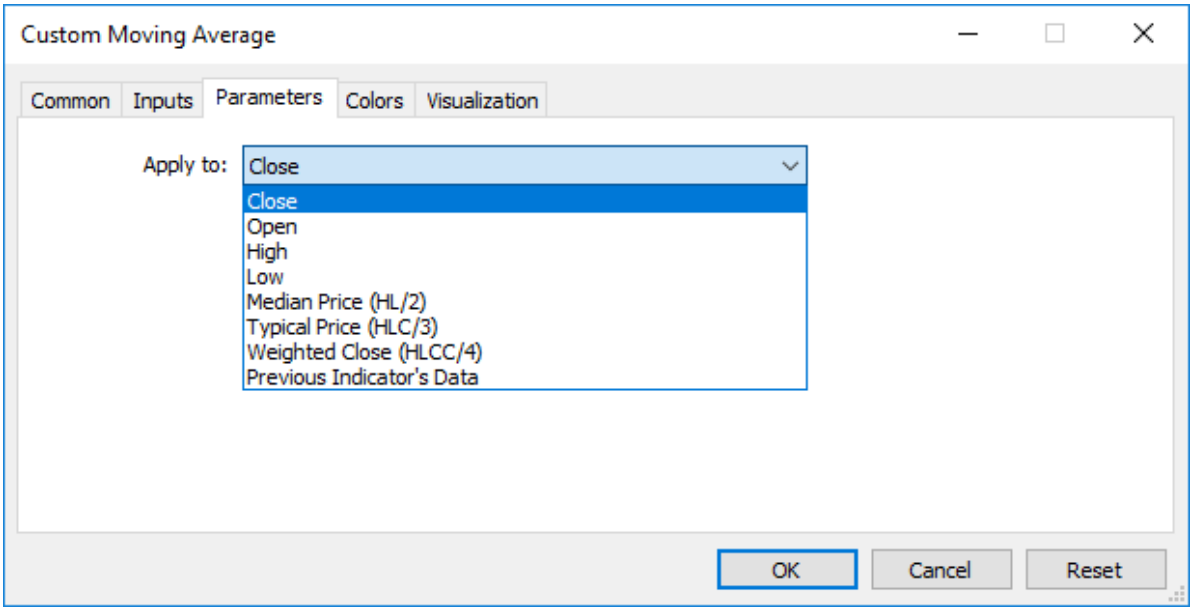
Um indicador personalizado deve ser compilado (com extensão EX5) e localizado no diretório MQL5/Indicadores do terminal do cliente ou seu subdiretório.

Indicadores que requerem teste são definidos automaticamente a partir da chamada da função iCustom(), se o correspondente parâmetro for definido através de um string constante. Para todos os outros casos (uso da função IndicatorCreate() ou uso de uma string não-constante no parâmetro de define o nome do indicador) a propriedade #property tester_indicator é requerida:

```
#property tester_indicator "indicator_name.ex5"
```

Se o primeiro formulário de chamada é usado no indicador, então na inicialização do indicador, na aba "Parâmetros", você pode escolher dados adicionais para cálculo. Se a opção "Aplicar para" do parâmetro não é selecionada, o cálculo padrão baseia-se nos valores "Fechamento" dos preços.

© 2000-2024, MetaQuotes Ltd.



Quando você chama um indicador personalizado a partir do programa MQL5, o parâmetro Applied_Price ou um manipulador de outro indicador deve ser passado por último, depois de todas as variáveis de entrada do indicador personalizado.

Também Veja

Propriedades de Programa, Séries Temporais e Acesso a Indicadores, IndicatorCreate(), IndicatorRelease()

Exemplo:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Plotar Etiqueta1
#property indicator_label1  "Etiqueta1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- parâmetros de entrada
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- buffers do indicador
double         Label1Buffer[];
//--- Manipulador do indicador personalizado Moving Average.mq5
int MA_handle;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- mapeamento de buffers do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // usando o fechamento de preços
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- Copiar os valores do indicador Custom Moving Average para o nosso buffer do ind
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- Se a nossa tentativa falhou - Reportar isto
   if(copy<=0)
      Print("Uma tentativa de obter os valores se houve falha do Custom Moving Average
//--- valor retorno de prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

iDEMA

A função retorna o manipulador do indicador Double Exponential Moving Average. Tem apenas um buffer.

```
int  iDEMA(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   int                 ma_shift,          // deslocamento horizontal
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio (barras contadas) para os cálculos.

ma_shift

[in] Deslocamento do indicador relativo ao gráfico de preços.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iDEMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
```

```
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iDEMA
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,             // usar iDEMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iDEMA;           // tipo de função
input int                  ma_period=14;              // período de média móvel
input int                  ma_shift=0;                // deslocamento
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iDEMABuffer[];
//--- variável para armazenar o manipulador do indicator iDEMA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Double Exponential Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[3];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de preço
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iDEMA para o símbolo %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Double Exponential Moving Average é calc
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iDEMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iDEMABuffer é maior do que o número de valores no indicador iDE
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iDEMABuffer com valores do indicador Double Exponential Moving
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Double Exponential Moving Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Preencher buffers do indicador a partir do indicador iDEMA       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Dou
                         int shift,         // deslocamento
                         int ind_handle,    // manipulador do indicador iDEMA
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iDEMABuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iDEMA, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iDeMarker

A função retorna o manipulador do indicador Double DeMarker. Tem apenas um buffer.

```
int  iDeMarker(
   string           symbol,        // símbolo nome
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio (barras contadas) para os cálculos.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iDeMarker."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iDeMarker
#property indicator_label1  "iDeMarker"\
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- níveis horizontais na janela de indicador
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // usar iDeMarker
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iDeMarker;       // tipo de função
input int                   ma_period=14;              // período de média móvel
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double          iDeMarkerBuffer[];
//--- variável para armazenar o manipulador do indicator iDeMarker
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador DeMarker
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
```

© 2000-2024, MetaQuotes Ltd.

```
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[1];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iDeMarker para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador DeMarker é calculado para
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iDeMarker
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
```

```
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iDeMarkerBuffer é maior do que o número de valores no indicador
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iDeMarkerBuffer com valores do indicador DeMarker
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador DeMarker
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iDeMarker   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do DeM
                         int ind_handle,    // manipulador do indicador iDeMarker
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iDeMarkerBuffer com valores do buffer do indicador
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iDeMarker, código de erro %d",Ge
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iEnvelopes

A função retorna o manipulador do indicador Envelopes.

```
int  iEnvelopes(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período para o cálculo da linha da média
   int                 ma_shift,          // deslocamento horizontal do indicador
   ENUM_MA_METHOD      ma_method,         // tipo de suavização
   ENUM_APPLIED_PRICE  applied_price,     // tipo de preço ou de manipulador
   double              deviation          // desvio dos limites da linha mediana (em p
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período da média móvel para a linha principal.

ma_shift

[in] O deslocamento do indicador em relação à gráfico de preço.

ma_method

[in] Tipo suavizado. Pode ser um dos valores ENUM_MA_METHOD.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

deviation

[in] O desvio da linha principal (em porcentagem).

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|                                                  Demo_iEnvelopes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                                ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- plotar linha superior
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar linha inferior
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // usar iEnvelopes
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iEnvelopes;      // tipo de função
input int                   ma_period=14;              // período de média móvel
input int                   ma_shift=0;                // deslocamento
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo de suavização
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de preço
input double                deviation=0.1;             // desvio das bordas da média mo
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         UpperBuffer[];
double         LowerBuffer[];
//--- variável para armazenar o manipulador do indicator iEnvelopes
```

```
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Envelopes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- definir o deslocamento de cada linha
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviat
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[5];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavização
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- tipo de preço
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iEnvelopes para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Envelopes é calculado para
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iEnvelopes
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- se o array UpperBuffer é maior do que o número de valores no indicador iEr
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays UpperBuffer e LowerBuffer com valores do indicador Envelopes
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Envelopes
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iEnvelopes  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // buffer do indicator linha da
                           double &lower_values[],    // indicador da linha da borda i
                           int shift,                 // deslocamento
                           int ind_handle,            // manipulador do indicador iEnv
                           int amount                 // número dos valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array UpperBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iEnvelopes, código de erro %d",G
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- preencher uma parte do array LowerBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iEnvelopes, código de erro %d",G
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iForce

A função retorna o manipulador do indicador Force Index. Tem apenas um buffer.

```
int  iForce(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   ENUM_MA_METHOD      ma_method,         // tipo suavizado
   ENUM_APPLIED_VOLUME applied_volume     // tipo de volume para cálculo
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio para cálculo do indicador.

ma_method

[in] Tipo suavizado. Pode ser um dos valores ENUM_MA_METHOD.

applied_volume

[in] O volume usado. Pode ser um dos valores ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iForce.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iForce."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- desenhando iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // usar iForce
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iForce;             // tipo de função
input int                  ma_period=13;                 // período médio
input ENUM_MA_METHOD       ma_method=MODE_SMA;           // tipo de suavização
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // tipo de volume
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iForceBuffer[];
//--- variável para armazenar o manipulador do indicator iForce
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Force
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- tomar o símbolo do gráfico, o indicador está anexado para
       name=_Symbol;
      }
//--- criar manipulador do indicador
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[3];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de suavização
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- tipo de volume
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- tipo de preço
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iForce para o símbolo %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Force é calculado para
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iForce
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iForceBuffer é maior do que o número de valores no indicador iF
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iForceBuffer com valores do indicador Force
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Force
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iForce      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Fo
```

```
                          int ind_handle,     // manipulador do indicador iForce
                          int amount          // número de valores copiados
                          )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iForceBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iForce, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iFractals

A função retorna o manipulador do indicador Fractals.

```
int  iFractals(
   string           symbol,      // símbolo nome
   ENUM_TIMEFRAMES  period       // período
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer são os seguintes: 0 - UPPER_LINE, 1 - LOWER_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iFractals.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iFractals."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- plotar FractalUp
```

```
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- plotar FractalDown
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,          // usar iFractals
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iFractals;          // tipo de função
input string                symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // timeframe
//--- buffers do indicador
double         FractalUpBuffer[];
double         FractalDownBuffer[];
//--- variável para armazenar o manipulador do indicator iFractals
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Fractals
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- definir códigos usando um símbolo do conjunto de caractres Wingdings para a prop
   PlotIndexSetInteger(0,PLOT_ARROW,217); // seta acima
   PlotIndexSetInteger(1,PLOT_ARROW,218); // seta abaixo
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iFractals para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Fractals é calculado para
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iFractals
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
```

```
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array FractalUpBuffer é maior do que o número de valores no indicador
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays FractalUpBuffer e FractalDownBuffer com valores do indicador
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Fractals
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iFractals   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // buffer do indicator para set
                           double &down_arrows[],      // buffer do indicator para set
                           int ind_handle,             // manipulador do indicador iFr
                           int amount                  // número de valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array FractalUpBuffer com valores do buffer do indicador
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iFractals ao array FractalUpBuff
                  GetLastError());
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
```

```
//--- preencher uma parte do array FractalDownBuffer com valores do buffer do indicado
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iFractals ao array FractalDownBu
                  GetLastError());
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iFrAMA

A função retorna o manipulador do indicador Fractal Adaptive Moving Average. Tem apenas um buffer.

```
int  iFrAMA(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   int                 ma_shift,          // deslocamento horizontal no gráfico
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período (contando as barras) para os cálculos do indicador.

ma_shift

[in] Deslocamento do indicador no gráfico de preço.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iFrAMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iFrAMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
```

```
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- desenhando iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // usar iFrAMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iFrAMA;             // tipo de função
input int                  ma_period=14;                 // período da média móvel
input int                  ma_shift=0;                   // deslocamento
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // tipo de preço
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iFrAMABuffer[];
//--- variável para armazenar o manipulador do indicator iFrAMA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Fractal Adaptive Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[3];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de preço
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- tipo de preço
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iFrAMA para o símbolo %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador iFrAMA é calculado para
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2024, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iFrAMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iFrAMABuffer é maior do que o número de valores no indicador iF
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iFrAMABuffer com valores do indicador Fractal Adaptive Moving
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Fractal Adaptive Moving Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Preencher buffers do indicador a partir do indicador iFrAMA      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Fra
                         int shift,         // deslocamento
                         int ind_handle,    // manipulador do indicador iFrAMA
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iFrAMABuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iFrAMA, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iGator

A função retorna o manipulador do indicador Gator. O oscilador mostra a diferença entre as linhas azul e vermelha do Jacaré (histograma superior) e diferença entre as linhas vermelha e verde (menor histograma).

```
int  iGator(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 jaw_period,        // período para cálculo da mandíbula
   int                 jaw_shift,         // deslocamento horizontal da mandíbula
   int                 teeth_period,      // período para cálculo dos dentes
   int                 teeth_shift,       // deslocamento horizontal dos dentes
   int                 lips_period,       // período para o cálculo da boca
   int                 lips_shift,        // deslocamento horizontal da boca
   ENUM_MA_METHOD      ma_method,         // tipo de suavização
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

jaw_period

[in] Período médio para a linha azul (mandíbula do Alligator).

jaw_shift

[in] O deslocamento da linha azul em relação à tabela de preços. Não está diretamente ligado ao deslocamento visual do histograma do indicador.

teeth_period

[in] Período médio para a linha vermelha (dentes do Alligator).

teeth_shift

[in] O deslocamento da linha vermelha em relação para a tabela de preços. Não está diretamente ligado ao deslocamento visual do histograma do indicador.

lips_period

[in] Período médio para a linha verde (boca de Alligator).

lips_shift

[in] O deslocamento da linha verde em relação à tabela de preços. Não está diretamente ligado ao deslocamento visual do histograma do indicador.

ma_method

[in] Tipo suavizado. Pode ser um dos valores ENUM_MA_METHOD.

© 2000-2024, MetaQuotes Ltd.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Números de buffer: 0 - UPPER_HISTOGRAM, 1 - buffer de cor do histograma superior, 2 - LOWER_HISTOGRAM, 3 - buffer de cor do histograma inferior.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iGator.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "dos buffers do indicador para o indicador técnico iGator."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares aos do padrão Gator Os

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- desenhando GatorUp
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- desenhando GatorDown
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
```

```
  {
   Call_iGator,            // usar iGator
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iGator;       // tipo de função
input string               symbol=" ";             // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // timeframe
input int                  jaw_period=13;          // período da linha da mandíbula
input int                  jaw_shift=8;            // deslocamento da linha da mandíbu
input int                  teeth_period=8;         // período da linha dos dentes
input int                  teeth_shift=5;          // deslocamento da linha dos dentes
input int                  lips_period=5;          // período da linha da boca
input int                  lips_shift=3;           // deslocamento da linha da boca
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // método das linhas médias do All
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// tipo de preço utilizado para
//--- buffers do indicador
double         GatorUpBuffer[];
double         GatorUpColors[];
double         GatorDownBuffer[];
double         GatorDownColors[];
//--- variável para armazenar o manipulador do indicator iGator
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- deslocamento dos valores para os histogramas superior e inferior
int shift;
//--- manteremos o número de valores no indicador Gator Oscillator
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  Todas as mudanças nos parâmetros especificados referem-se o indicador Alligator, que
  Isso é porque eles não se movem como o indicador Gator em si, mas eles se movem con
  Cujo os valores são utilizados para calcular o Gator Oscillator!
*/
//--- vamos calcular o deslocamento para os histogramas superior e inferior, que é igu
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
```

```
//--- apesar do indicador conter dois histogramas, o mesmo deslocamento é usado - está
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[8];
      //--- períodos e deslocamentos da linhas do Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- tipo de suavização
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- tipo de preço
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- criando o manipulador
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iGator para o símbolo %s/
```

© 2000-2024, MetaQuotes Ltd.

```
                    name,
                    EnumToString(period),
                    GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Gator Oscillator é calculado para
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iGator
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array GatorUpBuffer é maior do que o número de valores no indicador
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
```

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador Gator Oscillator
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Gator Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iGator      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // buffer do indicator para o
                           double &up_color_buffer[],    // buffer do indicator para o
                           double &downs_buffer[],       // buffer do indicator para o
                           double &downs_color_buffer[], // buffer do indicator para o
                           int u_shift,                  // deslocamento para o histog
                           int ind_handle,               // manipulador do indicador i
                           int amount                    // número de valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array GatorUpBuffer com valores do buffer do indicador qu
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iGator, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array GatorUpColors com valores do buffer do indicador qu
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iGator, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
```

```
//--- preencher uma parte do array GatorDownBuffer com valores do buffer do indicador
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iGator, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array GatorDownColors com valores do buffer do indicador
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iGator, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iIchimoku

A função retorna o manipulador do indicador Ichimoku Kinko Hyo.

```
int  iIchimoku(
   string           symbol,           // símbolo nome
   ENUM_TIMEFRAMES  period,           // período
   int              tenkan_sen,       // período Tenkan-sen
   int              kijun_sen,        // período Kijun-sen
   int              senkou_span_b     // período Senkou Span B
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

tenkan_sen

[in] Período médio para Tenkan Sen.

kijun_sen

[in] Período médio para Kijun Sen.

senkou_span_b

[in] Período médio para Senkou Span B.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
```

```
#property description "de buffers do indicador para o indicador técnico iIchimoku."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros, assim como no padrão Ichimoku Kinko

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- plotar Tenkan_sen
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar Kijun_sen
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plotar Senkou_Span
#property indicator_label3  "Senkou Span A;Senkou Span B" // two fields will be shown
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- plotar Chikou_Span
#property indicator_label4  "Chikou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,          // usar iIchimoku
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iIchimoku;       // tipo de função
input int                   tenkan_sen=9;              // período Tenkan-sen
input int                   kijun_sen=26;              // período Kijun-sen
input int                   senkou_span_b=52;          // período Senkou Span B
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
double         Senkou_Span_A_Buffer[];
double         Senkou_Span_B_Buffer[];
double         Chinkou_Span_Buffer[];
//--- variável para armazenar o manipulador do indicator iIchimoku
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Ichimoku Kinko Hyo
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- definir o deslocamento para o canal Senkou Span das barras kijun_sen na futura c
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- não é necessário definir um deslocamento para a linha Span Chinkou, uma vez que
//--- já estão armazenados com um deslocamento na iIchimoku
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[3];
      //--- períodos e deslocamentos da linhas do Alligator
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- criando o manipulador
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iIchimoku para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Ichimoku Kinko Hyo é calculado para
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iIchimoku
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
```

```
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array Tenkan_sen_Buffer é maior do que o número de valores no indicad
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador Ichimoku Kinko Hyo
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Ichimoku Kinko Hyo
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iIchimoku   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // buffer do indicador da
                           double &kijun_sen_buffer[],      // buffer do indicator da
                           double &senkou_span_A_buffer[],  // buffer do indicator da
                           double &senkou_span_B_buffer[],  // buffer do indicator da
                           double &chinkou_span_buffer[],   // buffer do indicator da
                           int senkou_span_shift,           // deslocamento das linhas
                           int ind_handle,                  // manipulador do iIchimok
                           int amount                       // número de valores copia
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array Tenkan_sen_Buffer com valores do buffer do indicado
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
```

```
      PrintFormat("1.Falha ao copiar dados do indicador iIchimoku, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array Kijun_sen_Buffer com valores do buffer do indicado
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("2.Falha ao copiar dados do indicador iIchimoku, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array Chinkou_Span_Buffer com valores do buffer do indica
//--- se senkou_span_shift>0, a linha é deslocada na direção futura pelas barras senko
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("3.Falha ao copiar dados do indicador iIchimoku, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array com valores do buffer do indicador que tem índice 1
//--- se senkou_span_shift>0, a linha é deslocada na direção futura pelas barras senko
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("4.Falha ao copiar dados do indicador iIchimoku, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array Senkou_Span_B_Buffer com valores do buffer do indic
//--- ao copiar Chinkou Span, não precisamos considerar a mudança, uma vez que os dado
//--- já estão armazenados com um deslocamento na iIchimoku
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("5.Falha ao copiar dados do indicador iIchimoku, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iBWMFI

A função retorna o manipulador do indicador Market Facilitation Index. Tem apenas um buffer.

```
int  iBWMFI(
   string               symbol,             // símbolo nome
   ENUM_TIMEFRAMES      period,             // período
   ENUM_APPLIED_VOLUME  applied_volume      // tipo de volume para o cálculo
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

applied_volume

[in] O volume usado. Pode ser uma das constantes do ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iBWMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iBWMFI."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar iBWMFI
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,            // usar iBWMFI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iBWMFI;          // tipo de função
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;// tipo de volume
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iBWMFIBuffer[];
double         iBWMFIColors[];
//--- variável para armazenar o manipulador do indicator iBWMFI
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- variável para armazenar o manipulador do indicador Market Facilitation Index po
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
```

```
      {
       //--- preencher a estrutura com os parâmetros do indicador
       MqlParam pars[1];
       //--- tipo de volume
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
      }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
      {
       //--- mensagem sobre a falha e a saída do código de erro
       PrintFormat("Falha ao criar o manipulador do indicador iBWMFI para o símbolo %s/
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- o indicador é interrompido precocemente
       return(INIT_FAILED);
      }
//--- mostrar que o símbolo/período do indicador Market Facilitation Index por Bill W
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiado do indicador iBWMFI
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
       return(0);
      }
```

```
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iBWMFIBuffer é maior do que o número de valores no indicador iB
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com os valores do indicador Market Facilitation Index por Bi
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Market Facilitation Index por Bill Wi
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iBWMFI      |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // buffer do indicator para os valores
                           double &colors[],    // buffer do indicator para cores do h
                           int ind_handle,      // manipulado do indicador iBWMFI
                           int amount           // número de valores copiado
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iBWMFIBuffer com valores do indicador buffer do ind
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iBWMFI, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
```

```
//--- preencher uma parte do array iBWMFIColors com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iBWMFI, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMomentum

A função retorna o manipulador do indicador Momentum. Tem apenas um buffer.

```
int  iMomentum(
   string                symbol,            // símbolo nome
   ENUM_TIMEFRAMES       period,            // período
   int                   mom_period,        // período médio
   ENUM_APPLIED_PRICE    applied_price      // tipo de preço ou de manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

mom_period

[in] Período médio(barras contadas) para o cálculo de modificação do preço.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iMomentum."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Momentum."

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iMomentum
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,         // usar iMomentum
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iMomentum;       // tipo de função
input int                  mom_period=14;             // período do indicador Momentum
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iMomentumBuffer[];
//--- variável para armazenar o manipulador do indicator iMomentum
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Momentum
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- criar manipulador do indicador
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- tipo de preço
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iMomentum para o símbolo
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Momentum é calculado para
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iMomentum
```

© 2000-2024, MetaQuotes Ltd.

```
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iMomentumBuffer é maior do que o número de valores no indicador
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iMomentumBuffer com valores do indicador Momentum
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Momentum
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iMomentum   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Mom
                         int ind_handle,    // manipulador do indicador iMomentum
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iMomentumBuffer com valores do buffer do indicador
```

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iMomentum, código de erro %d",Ge
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMFI

A função retorna o manipulador do indicador Money Flow Index.

```
int  iMFI(
   string                symbol,              // símbolo nome
   ENUM_TIMEFRAMES       period,              // período
   int                   ma_period,           // período médio
   ENUM_APPLIED_VOLUME   applied_volume       // tipo de volume para o cálculo
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio (barras contadas) para cálculo.

applied_volume

[in] O volume usado. Pode ser um dos valores ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMFI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iMFI."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Money Flow I

#property indicator_separate_window
#property indicator_buffers 1
```

```
#property indicator_plots   1
//--- plotar iMFI
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- níveis horizontais na janela de indicador
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // usar iMFI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iMFI;               // tipo de função
input int                  ma_period=14;                 // período
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // tipo de volume
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iMFIBuffer[];
//--- variável para armazenar o manipulador do indicator iMFI
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Money Flow Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de volume
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador para o símbolo %s/%s, cód
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Money Flow Index é calculado para
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- número de valores copiados a partir do indicador iMFI
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iMFIBuffer é maior do que o número de valores no indicador iMFI
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iMFIBuffer com valores do indicador Money Flow Index
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Flow Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iMFI        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // buffer do indicator para valores do Mon
                         int ind_handle,    // manipulador do indicador iMFI
                         int amount         // número de valores copiados
                         )
  {
//--- redefinir o código de erro
```

```
   ResetLastError();
//--- preencher uma parte do array iMFIBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iMFI, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMA

A função retorna o manipulador do indicador Moving Average. Tem apenas um buffer.

```
int  iMA(
   string                symbol,            // símbolo nome
   ENUM_TIMEFRAMES       period,            // período
   int                   ma_period,         // período médio
   int                   ma_shift,          // deslocamento horizontal
   ENUM_MA_METHOD        ma_method,         // tipo suavizado
   ENUM_APPLIED_PRICE    applied_price      // tipo de preço ou de manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio para cálculo da média móvel.

ma_shift

[in] Deslocamento do indicador relativo ao gráfico de preços.

ma_method

[in] Tipo suavizado. Pode ser um dos valores ENUM_MA_METHOD.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros, assim como no padrão Moving Average

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iMA
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,                // usar iMA
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iMA;                // tipo de função
input int                  ma_period=10;                 // período da média móvel
input int                  ma_shift=0;                   // deslocamento
input ENUM_MA_METHOD       ma_method=MODE_SMA;           // tipo de suavização
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // tipo de preço
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iMABuffer[];
//--- variável para armazenar o manipulador do indicator iMA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavização
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iMA para o símbolo %s/%s,
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Moving Average é calculado para
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iMABuffer é maior do que o número de valores no indicador iMA p
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array iMABuffer com valores do indicador Adaptive Moving Average
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Moving Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador MA          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // buffer do indicator para valores do Mo
                         int shift,         // deslocamento
                         int ind_handle,     // manipulador do indicador iMA
                         int amount          // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iMABuffer com valores do buffer do indicador que te
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iMA, código de erro %d",GetLastE
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iOsMA

A função retorna o manipulador do indicador Moving Average of Oscillator. O oscilador OsMA mostra a diferença entre valores do indicador MACD e sua linha de sinal. Tem apenas um buffer.

```
int  iOsMA(
   string              symbol,              // símbolo nome
   ENUM_TIMEFRAMES     period,              // período
   int                 fast_ema_period,     // período Média Móvel Rápida
   int                 slow_ema_period,     // período Média Móvel Lenta
   int                 signal_period,       // período para diferença entre as médias
   ENUM_APPLIED_PRICE  applied_price        // tipo de preço ou de manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

fast_ema_period

[in] Período para cálculo da Média Móvel Rápida.

slow_ema_period

[in] Período para cálculo da Média Móvel Lenta.

signal_period

[in] Período para o cálculo da linha de sinal.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Em alguns sistemas este oscilador é também conhecido como histograma MACD.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOsMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iOsMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Moving Avera

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iOsMA
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // usar iOsMA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iOsMA;           // tipo de função
input int                  fast_ema_period=12;        // período da Média Móvel Rápida
input int                  slow_ema_period=26;        // período da Média Móvel Lenta
input int                  signal_period=9;           // período da diferença entre as
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iOsMABuffer[];
//--- variável para armazenar o manipulador do indicador iAMA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
```

```
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período da média móvel rápida
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- período lento da média móvel
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- período médio da diferença entre a média móvel rápida e a lenta
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iOsMA para o símbolo %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Moving Average of Oscillator é calculado
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
```

© 2000-2024, MetaQuotes Ltd.

```
                              fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iOsMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iOsMABuffer é maior do que o número de valores no indicador iOs
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador iOsMA
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
```

© 2000-2024, MetaQuotes Ltd.

```
                              short_name,
                              values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Moving Average of Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iOsMA       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iOsMA
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iOsMABuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iOsMA, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iMACD

A função retorna o manipulador do indicador Moving Averages Convergence/Divergence. Em sistemas onde OsMA é chamado Histograma MACD, este indicador é apresentado com duas linhas. No terminal do cliente, o indicador Moving Averages Convergence/Divergence é apresentado como um histograma.

```
int  iMACD(
   string              symbol,              // símbolo nome
   ENUM_TIMEFRAMES     period,              // período
   int                 fast_ema_period,     // período para cálculo da média móvel rá
   int                 slow_ema_period,     // período para cálculo da média móvel le
   int                 signal_period,       // período para diferença entre as médias
   ENUM_APPLIED_PRICE  applied_price        // tipo de preço ou de manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

fast_ema_period

[in] Período para cálculo da Média Móvel Rápida.

slow_ema_period

[in] Período para cálculo da Média Móvel Lenta.

signal_period

[in] Período para cálculo da linha de Sinal.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer são os seguintes: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMACD.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
```

```
//|                                                  ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iMACD."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros, assim como no padrão MACD."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- plotar MACD
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // usar iMACD
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iMACD;           // tipo de função
input int                  fast_ema_period=12;        // período da Média Móvel Rápida
input int                  slow_ema_period=26;        // período da Média Móvel Lenta
input int                  signal_period=9;           // período da diferença entre as
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffers do indicador
double         MACDBuffer[];
double         SignalBuffer[];
//--- variável para armazenar o manipulador do indicator iMACD
int    handle;
//--- variável para armazenamento
```

```
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Moving Averages Convergence/Diverger
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período da média móvel rápida
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- período lento da média móvel
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- período médio da diferença entre a média móvel rápida e a lenta
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("Falha ao criar o manipulador do indicador iMACD para o símbolo %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Moving Average Convergence/Divergence é
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iMACD
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array MACDBuffer é maior do que o número de valores no indicador iMAC
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
```

```
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador iMACD
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Moving Averages Convergence/Divergenc
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iMACD       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // buffer do indicator para valor
                           double &signal_buffer[],  // buffer do indicator da linha c
                           int ind_handle,           // manipulador do indicador iMACD
                           int amount                // número de valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iMACDBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iMACD, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }

//--- preencher uma parte do array SignalBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iMACD, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iOBV

A função retorna o manipulador do indicador On Balance Volume. Tem apenas um buffer.

```
int  iOBV(
   string                 symbol,             // símbolo nome
   ENUM_TIMEFRAMES        period,             // período
   ENUM_APPLIED_VOLUME    applied_volume      // tipo de volume para cálculo
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

applied_volume

[in] O volume usado. Pode ser um dos valores ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iOBV.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iOBV."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // usar iOBV
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iOBV;                // tipo de função
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;    // tipo de volume
input string               symbol=" ";                    // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;         // timeframe
//--- buffers do indicador
double         iOBVBuffer[];
//--- variável para armazenar o manipulador do indicator iOBV
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador On Balance Volume
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
```

© 2000-2024, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- tipo de volume
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador para o símbolo %s/%s, cód
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador On Balance Volume é calculado para
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iOBV
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iOBVBuffer é maior do que o número de valores no indicador iOBV
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador iOBV
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador On Balance Volume
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iOBV        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iOBV
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iOBVBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iOBV, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iSAR

A função retorna o manipulador do indicador Parabolic Stop and Reverse system. Tem apenas um buffer.

```
int  iSAR(
   string           symbol,       // símbolo nome
   ENUM_TIMEFRAMES  period,       // período
   double           step,         // incrementar passo
   double           maximum       // máximo nível de parada
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

step

[in] O nível de parada incrementado, usualmente 0.02.

maximum

[in] O máximo nível de parada, usualmente 0.2.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iSAR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iSAR."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Parabolic St

#property indicator_chart_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- desenhando iSAR
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // usar iSAR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iSAR;               // tipo de função
input double               step=0.02;                    // passo - o fator de acelera
input double               maximum=0.2;                  // máximo valor do passo
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffers do indicador
double         iSARBuffer[];
//--- variável para armazenar o manipulador do indicator iSAR
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Parabolic SAR
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- definir um código de símbolo do conjunto de caracteres Wingdings para a propriec
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
```

```
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- valor do passo
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- limite do valor do passo que pode ser usado para os cálculos
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iSAR para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Parabolic SAR é calculado para
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- número de valores copiados a partir do indicador iSAR
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iSARBuffer é maior do que o número de valores no indicador iSAR
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador iSAR
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador SAR
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iSAR        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iSAR
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
```

```
   ResetLastError();
//--- preencher uma parte do array iSARBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iSAR, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iRSI

A função retorna o manipulador do indicador Relative Strength Index. Tem apenas um buffer.

```
int  iRSI(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio para cálculo do RSI.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iRSI."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Relative St

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- desenhando iRSI
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- limites para mostrar os valores na janela do indicador
#property indicator_maximum 100
#property indicator_minimum 0
//--- níveis horizontais na janela de indicador
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // usar iRSI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iRSI;               // tipo de função
input int                  ma_period=14;                 // período da média móvel
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;    // tipo de preço
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffer do indicador
double         iRSIBuffer[];
//--- variável para armazenar o manipulador do indicator iRSI
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Relative Strength Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- período de média móvel
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- limite do valor do passo que pode ser usado para os cálculos
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iRSI para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Relative Strength Index é calculado para
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iRSI
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iRSIBuffer é maior do que o número de valores no indicador iRSI
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador iRSI
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Relative Strength Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador             |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iRSI
```

```
                          int amount              // número de valores copiados
                          )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iRSIBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iRSI, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iRVI

A função retorna o manipulador do indicador Relative Vigor Index.

```
int  iRVI(
   string           symbol,        // símbolo nome
   ENUM_TIMEFRAMES  period,        // período
   int              ma_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio para cálculo do indicador RVI.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

Os números de buffer são os seguintes: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iRVI."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Relative Vig

#property indicator_separate_window
#property indicator_buffers 2
```

```
#property indicator_plots   2
//--- plotar RVI
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // usar iRVI
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iRVI;               // tipo de função
input int                  ma_period=10;                 // período para cálculos
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffers do indicador
double         RVIBuffer[];
double         SignalBuffer[];
//--- variável para armazenar o manipulador do indicator iRVI
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Relative Vigor Index
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
```

© 2000-2024, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[1];
      //--- período para cálculos
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iRVI para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Relative Vigor Index é calculado para
   short_name=StringFormat("iRVI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2024, MetaQuotes Ltd.

```
  {
//--- número de valores copiados a partir do indicador iRVI
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array RVIBuffer é maior do que o número de valores no indicador iRVI
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador iRVI
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Relative Vigor Index
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iRVI        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // buffer do indicator para valores
                         double &signal_buffer[],  // buffer do indicator da linha de
                         int ind_handle,           // manipulador do indicador iRVI
                         int amount                // número de valores copiados
                         )
  {
```

```
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iRVIBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iRVI, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- preencher uma parte do array SignalBuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iRVI, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iStdDev

A função retorna o manipulador do indicador Standard Deviation. Tem apenas um buffer.

```
int  iStdDev(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   int                 ma_shift,          // deslocamento horizontal
   ENUM_MA_METHOD      ma_method,         // tipo suavizado
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio para cálculos do indicador.

ma_shift

[in] Deslocamento do indicador relativo ao gráfico de preços.

ma_method

[in] Tipo de média. Pode ser qualquer um dos valores do ENUM_MA_METHOD.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iStdDev.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iStdDev."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Deviation."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iStdDev
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,            // usar iStdDev
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iStdDev;          // tipo de função
input int                   ma_period=20;               // período médio
input int                   ma_shift=0;                 // deslocamento
input ENUM_MA_METHOD        ma_method=MODE_SMA;         // tipo de suavização
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;  // tipo de preço
input string                symbol=" ";                 // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // timeframe
//--- buffer do indicador
double         iStdDevBuffer[];
//--- variável para armazenar o manipulador do indicator iStdDev
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Standard Deviation
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de suavização
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iStdDev para o símbolo %s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Standard Deviation é calculado para
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iStdDev
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iStdDevBuffer é maior do que o número de valores no indicador i
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Standard Deviation
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Standard Deviation
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iStdDev     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // buffer do indicator para linha Star
                         int std_shift,         // deslocamento da linha Standard Devi
                         int ind_handle,        // manipulador do indicador iStdDev
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iStdDevBuffer com valores do buffer do indicador qu
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iStdDev, código de erro %d",GetL
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iStochastic

A função retorna o manipulador do indicador Stochastic Oscillator.

```
int  iStochastic(
   string           symbol,          // símbolo nome
   ENUM_TIMEFRAMES  period,          // período
   int              Kperiod,         // K-period (número de barras para cálculos)
   int              Dperiod,         // D-period (período da primeira suavização)
   int              slowing,         // final da suavização
   ENUM_MA_METHOD   ma_method,       // tipo de suavização
   ENUM_STO_PRICE   price_field      // método de cálculo estocástico
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

Kperiod

[in] Período médio (barras contadas) para o cálculo da linha K%.

Dperiod

[in] Período médio (barras contadas) para o cálculo da linha %D.

slowing

[in] Valor da desaceleração.

ma_method

[in] Tipo de média. Pode ser qualquer um dos valores do ENUM_MA_METHOD.

price_field

[in] Parâmetros de seleção de preço para cálculos. Pode ser um dos valoresENUM_STO_PRICE.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Observação

O número de buffer: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                          Demo_iStochastic.mq5 |
```

© 2000-2024, MetaQuotes Ltd.

```
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iStochastic."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- plotar Stochastic
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plotar Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- definir o limite dos valores do indicador
#property indicator_minimum 0
#property indicator_maximum 100
//--- níveis horizontais na janela de indicador
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,        // usar iStochastic
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iStochastic;     // tipo de função
input int                   Kperiod=5;                 // o período K ( o número de ba
input int                   Dperiod=3;                 // o período D (o período da sua
input int                   slowing=3;                 // período final da suavização
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // tipo de suavização
input ENUM_STO_PRICE        price_field=STO_LOWHIGH;   // método de cálculo do Estocás
```

```
input string                symbol=" ";                 // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // timeframe
//--- buffers do indicador
double         StochasticBuffer[];
double         SignalBuffer[];
//--- variável para armazenar o manipulador do indicator iStochastic
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Stochastic Oscillatorr
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de arrays para buffers do indicador
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[5];
      //--- o período K para cálculos
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- o período D para suavização primária
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- o período K final para suavização
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- tipo de suavização
```

© 2000-2024, MetaQuotes Ltd.

```
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- método de cálculo do Estocástico
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador para o símbolo %s/%s, cód
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Stochastic Oscillator é calculado para
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iStochastic
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
```

```
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array StochasticBuffer é maior do que o número de valores no indicado
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador iStochastic
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Stochastic Oscillator
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iStochastic |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // buffer do indicador dos valore
                           double &signal_buffer[],  // buffer do indicator da linha d
                           int ind_handle,           // manipulador do indicador iStoc
                           int amount                // número de valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array StochasticBuffer com valores do buffer do indicador
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iStochastic, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- preencher uma parte do array SignalBuffer com valores do buffer do indicador que
```

```
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iStochastic, código de erro %d",
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iTEMA

A função retorna o manipulador do indicador Triple Exponential Moving Average. Tem apenas um buffer.

```
int  iTEMA(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   int                 ma_shift,          // deslocamento horizontal indicador
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio (barras contadas) para cálculo.

ma_shift

[in] Deslocamento do indicador relativo à tabela de preços.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTEMA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iTEMA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
```

```
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Triple Expon

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iTEMA
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,              // usar iTEMA
   Call_IndicatorCreate     // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation              type=Call_iTEMA;           // tipo de função
input int                   ma_period=14;              // período médio
input int                   ma_shift=0;                // deslocamento
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // tipo de preço
input string                symbol=" ";                // símbolo
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iTEMABuffer[];
//--- variável para armazenar o manipulador do indicator Triple Exponential Moving Ave
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Triple Exponential Moving Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
```

```
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[3];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- deslocamento
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- tipo de preço
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iTEMA para o símbolo %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Triple Exponential Moving Average é calc
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

© 2000-2024, MetaQuotes Ltd.

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iTEMA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iTEMABuffer é maior do que o número de valores no indicador iTE
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Triple Exponential Moving Average
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Triple Exponential Moving Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iTEMA       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // buffer do indicator para valores do
                         int t_shift,           // deslocamento da linha
                         int ind_handle,        // manipulador do indicador iTEMA
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iTEMABuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iTEMA, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iTriX

A função retorna o manipulador do indicador Triple Exponential Moving Averages Oscillator. Tem apenas um buffer.

```
int  iTriX(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 ma_period,         // período médio
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

ma_period

[in] Período médio (barras contadas) para cálculos.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTriX.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iTriX."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,             // usar iTriX
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iTriX;           // tipo de função
input int                  ma_period=14;              // período
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iTriXBuffer[];
//--- variável para armazenar o manipulador do indicator iTriX
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Triple Exponential Moving Averages (
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
//--- criar manipulador do indicador
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[2];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- tipo de preço
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iTriX para o símbolo %s/%
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Triple Exponential Moving Averages Oscil
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iTriX
```

© 2000-2024, MetaQuotes Ltd.

```
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iTriXBuffer é maior do que o número de valores no indicador iTr
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Triple Exponential Moving Averages Os
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Triple Exponential Moving Averages Os
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iTriX       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iTriX
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iTriXBuffer com valores do buffer do indicador que
```

```
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iTriX, código de erro %d",GetLas
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iWPR

A função retorna o manipulador do indicador Larry Williams' Percent Range. Tem apenas um buffer.

```
int  iWPR(
   string           symbol,          // símbolo nome
   ENUM_TIMEFRAMES  period,          // período
   int              calc_period      // período médio
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

calc_period

[in] Período (barras contadas) para cálculo do indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iWPR.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iWPR."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- definir o limite dos valores do indicador
#property indicator_minimum -100
#property indicator_maximum 0
//--- níveis horizontais na janela de indicador
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // usar iWPR
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iWPR;            // tipo de função
input int                  calc_period=14;            // período
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iWPRBuffer[];
//--- variável para armazenar o manipulador do indicator iWPR
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Larry Williams' Percent Range
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- criar manipulador do indicador
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[1];
      //--- período
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iWPR para o símbolo %s/%s
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Williams' Percent Range é calculado para
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iWPR
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iWPRBuffer é maior do que o número de valores no indicador iWPR
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Williams' Percent Range
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Larry Williams' Percent Range
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iWPR        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // buffer do indicator para valores do
                         int ind_handle,        // manipulador do indicador iWPR
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iWPRBuffer com valores do buffer do indicador que t
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iWPR, código de erro %d",GetLast
      //--- parar com resultado zero - significa que indicador é considerado como não
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iVIDyA

A função retorna o manipulador do indicador Variable Index Dynamic Average. Tem apenas um buffer.

```
int  iVIDyA(
   string              symbol,            // símbolo nome
   ENUM_TIMEFRAMES     period,            // período
   int                 cmo_period,        // período para indicador Chande Momentum
   int                 ema_period,        // período suavizado do indicador EMA
   int                 ma_shift,          // deslocamento horizontal no gráfico
   ENUM_APPLIED_PRICE  applied_price      // tipo de preço ou manipulador
   );
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

cmo_period

[in] Período (barras contadas) para cálculo do indicador Chande Momentum Oscillator.

ema_period

[in] Período do indicador EMA (barras contadas) para cálculo do fator de suavização.

ma_shift

[in] Deslocamento do indicador relativo ao gráfico de preços.

applied_price

[in] O preço usado. Pode ser qualquer das constantes de preços ENUM_APPLIED_PRICE ou um manipulador de outro indicador.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iVIDyA.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iVIDyA."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme
#property description "Todos os outros parâmetros são similares ao padrão Variable Inc

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plotar iVIDyA
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // usar iVIDyA
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iVIDyA;          // tipo de função
input int                  cmo_period=15;             // o período do indicador Chande
input int                  ema_period=12;             // período do fator de suavizaçã
input int                  ma_shift=0;                // deslocamento
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // tipo de preço
input string               symbol=" ";                // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // timeframe
//--- buffer do indicador
double         iVIDyABuffer[];
//--- variável para armazenar o manipulador do indicator iVIDyA
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Variable Index Dynamic Average
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
```

```
//--- definir deslocamento
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- preencher a estrutura com os parâmetros do indicador
      MqlParam pars[4];
      //--- o período do indicador Chande Momentum
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- período do fator de suavização
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- deslocamento
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- tipo de preço
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
     {
      //--- mensagem sobre a falha e a saída do código de erro
      PrintFormat("Falha ao criar o manipulador do indicador iVIDyA para o símbolo %s/
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- o indicador é interrompido precocemente
      return(INIT_FAILED);
     }
//--- mostra que o símbolo/prazo do indicador Variable Index Dynamic Average é calcula
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price))
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador iVIDyA
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
      return(0);
     }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iVIDyABuffer é maior do que o número de valores no indicador iV
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher o array com valores do indicador Variable Index Dynamic Average
//--- se FillArrayFromBuffer retorna falso, significa que a informação não está pronta
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Variable Index Dynamic Average
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iVIDyA      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// buffer do indicator para valores d
                         int v_shift,           // deslocamento da linha
                         int ind_handle,        // manipulador do indicador iVIDyA
                         int amount             // número de valores copiados
                         )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iVIDyABuffer com valores do buffer do indicador que
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iVIDyA, código de erro %d",GetLa
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

iVolumes

A função retorna o manipulador do indicador. Tem apenas um buffer.

```
int  iVolumes(
   string                symbol,             // símbolo nome
   ENUM_TIMEFRAMES       period,             // período
   ENUM_APPLIED_VOLUME   applied_volume      // tipo de volume para o cálculo
   )
```

Parâmetros

symbol

[in] O nome do símbolo de segurança, os dados que devem ser usados para calcular o indicador. O valor NULL significa o símbolo atual.

period

[in] O valor do período pode ser um dos valores ENUM_TIMEFRAMES, 0 (zero) significa o prazo corrente.

applied_volume

[in] O volume utilizado. Pode ser um dos valores ENUM_APPLIED_VOLUME.

Valor de retorno

Retorna o manipulador de um indicador técnico especifico, em caso de falha de retorna INVALID_HANDLE. A memória do computador pode ser liberada a partir de um indicador que não é mais utilizado, usando a função IndicatorRelease(), onde o manipulador de indicador é transmitido.

Exemplo:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iVolumes.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             ;https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "O indicador demonstra como obter dados"
#property description "de buffers do indicador para o indicador técnico iVolumes."
#property description "Um símbolo e o prazo utilizado para o cálculo do indicador,"
#property description "são definidos pelos parâmetros de símbolo e período."
#property description "O método de criação do manipulador é definido através do parâme

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotar iVolumes
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
```

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| Enumerador dos métodos de criação do manipulador                 |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // usar iVolumes
   Call_IndicatorCreate    // usar IndicatorCreate
  };
//--- parâmetros de entrada
input Creation             type=Call_iVolumes;           // tipo de função
input ENUM_APPLIED_VOLUME  applied_volume=VOLUME_TICK;   // tipo de volume
input string               symbol=" ";                   // símbolo
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // timeframe
//--- buffers do indicador
double         iVolumesBuffer[];
double         iVolumesColors[];
//--- variável para armazenar o manipulador do indicator iVolumes
int    handle;
//--- variável para armazenamento
string name=symbol;
//--- nome do indicador num gráfico
string short_name;
//--- manteremos o número de valores no indicador Volumes
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Função de inicialização do indicador customizado                 |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- atribuição de array para buffer do indicador
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- determinar o símbolo do indicador, é desenhado para
   name=symbol;
//--- excluir os espaços à direita e à esquerda
   StringTrimRight(name);
   StringTrimLeft(name);
//--- se resulta em comprimento zero da string do 'name'
   if(StringLen(name)==0)
     {
      //--- tomar o símbolo do gráfico, o indicador está anexado para
      name=_Symbol;
     }
//--- criar manipulador do indicador
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- preencher a estrutura com os parâmetros do indicador
       MqlParam pars[1];
       //--- tipo de preço
       pars[0].type=TYPE_INT;
       pars[0].integer_value=applied_volume;
       handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
      }
//--- se o manipulador não é criado
   if(handle==INVALID_HANDLE)
      {
       //--- mensagem sobre a falha e a saída do código de erro
       PrintFormat("Falha ao criar o manipulador do indicador iVolumes para o símbolo %
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- o indicador é interrompido precocemente
       return(INIT_FAILED);
      }
//--- mostra que o símbolo/prazo do indicador Volumes é calculado para
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- inicialização normal do indicador
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de iteração do indicador customizado                      |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- número de valores copiados a partir do indicador
   int values_to_copy;
//--- determinar o número de valores calculados no indicador
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
      {
       PrintFormat("BarsCalculated() retornando %d, código de erro %d",calculated,GetLa
       return(0);
      }
//--- se for o princípio do cálculo do indicador, ou se o número de valores é modifica
```

```
//--- ou se é necessário cálculo do indicador para duas ou mais barras (isso significa
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- se o array iVolumesBuffer é maior do que o número de valores no indicador
      //--- caso contrário, copiamos menor do que o tamanho dos buffers do indicador
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- isso significa que não é a primeira vez do cálculo do indicador, é desde a
      //--- para o cálculo não mais do que uma barra é adicionada
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- preencher os arrays com valores do indicador iVolumes
//--- se FillArraysFromBuffer retorna falso, isto significa que a informação ainda não
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- formar a mensagem
   string comm=StringFormat("%s ==>  Valor atualizado no indicador %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- exibir a mensagem de serviço no gráfico
   Comment(comm);
//--- memorizar o número de valores no indicador Volumes
   bars_calculated=calculated;
//--- retorna o valor prev_calculated para a próxima chamada
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Preencher buffers do indicador a partir do indicador iVolumes    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // buffer do indicator para valo
                           double &color_buffer[],    // buffer do indicator de cores
                           int ind_handle,            // manipulador do indicador iVol
                           int amount                 // número dos valores copiados
                           )
  {
//--- redefinir o código de erro
   ResetLastError();
//--- preencher uma parte do array iVolumesBuffer com valores do buffer do indicador c
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iVolumes, código de erro %d",Get
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- preencher uma parte do array iVolumesColors com valores do buffer do indicador c
```

```
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- Se a cópia falhar, informe o código de erro
      PrintFormat("Falha ao copiar dados do indicador iVolumes, código de erro %d",Get
      //--- parar com resultado zero - significa que indicador é considerado como não
      return(false);
     }
//--- está tudo bem
   return(true);
  }
//+------------------------------------------------------------------+
//| Função de desinicialização do indicador                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- limpar o gráfico após excluir o indicador
   Comment("");
  }
```

© 2000-2024, MetaQuotes Ltd.

Trabalhando com Resultados de Otimização

Funções para organizar processamento personalizado da otimização resultando no testador de estratégia. Elas podem ser chamadas durante a otimização de agentes de teste, bem como localmente em Expert Advisors e scripts.

Quando você executa um Expert Advisor no testador de estratégia, você pode criar seu próprio conjunto de dados com base nos tipos simples ou estruturas simples (não contêm strings, objetos de classe ou objetos de arrays dinâmicas). Este conjunto de dados podem ser salvos utilizando a função FrameAdd() numa estrutura especial chamando um quadro. Durante a otimização de um Expert Advisor cada agente pode enviar uma série de quadros para o terminal. Todos os quadros recebidos são gravados no arquivo *.MQD na pasta terminal_directory/MQL5/Files/Tester expert nomeada igual ao Expert Advisor. Eles são escritos na ordem em que são recebidas dos agentes. O recebimento de um quadro no terminal do cliente a partir de um agente de teste gera o evento TesterPass.

Os quadros podem ser armazenados na memória do computador e num arquivo com o nome especifico. A linguagem MQL5 não estabelece limitações sobre o número de quadros.

Limitações de memória e de espaço em disco na MQL5 Cloud Network

Ao executar a otimização na MQL5 Cloud Network, há uma restrição: o Expert Advisor testado não pode gravar mais de 4 GB de informações no disco e usar mais de 4 GB de RAM. Se o limite for excedido, o agente de rede não conseguirá concluir o cálculo corretamente e você não receberá o resultado do teste. Além disso, você será cobrado pelo tempo já gasto nos cálculos.

Se você precisar obter informações de cada passagem da otimização, use enviar quadros sem gravar no disco. Para não usar operações de arquivo nos Expert Advisors durante os cálculos na MQL5 Cloud Network, você pode aplicar essa verificação:

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

Função	Ação
FrameFirst	Move um ponteiro de leitura de quadro para o início e redefine previamente um conjunto de filtros
FrameFilter	Define a estrutura do filtro de leitura e move o ponteiro para o início
FrameNext	Lê um quadro e move o ponteiro para o próximo

© 2000-2024, MetaQuotes Ltd.

Função	Ação
FrameInputs	Recebe parâmetros de entrada, no qual o quadro é formado
FrameAdd	Adiciona um quadro com dados
ParameterGetRange	Recebe os dados na faixa de valores e o passo de alteração para uma variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia
ParameterSetRange	Especifica o uso de variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia: valor, mudança de passo, os valores iniciais e finais

Também Veja

Testando Estatísticas, Propriedades de Execução do Programa MQL5

© 2000-2024, MetaQuotes Ltd.

FrameFirst

Move um ponteiro de leitura de quadro para o início e redefine um conjunto de filtros.

```
bool  FrameFirst();
```

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

FrameFilter

Define a estrutura do filtro de leitura e move o ponteiro para o início.

```
bool  FrameFilter(
   const string  name,          // Nome público/etiqueta
   long          id             // ID Pública
   );
```

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

Observação

Se uma string vazia é passado como o primeiro parâmetro, o filtro funcionará apenas com um parâmetro numérico, ou seja, apenas quadros com o id especifico serão vistos. Se o valor do segundo parâmetro é ULONG_MAX, apenas o filtro de texto funcionará.

Chamar FrameFilter("", ULONG_MAX) é equivalente a chamar FrameFirst(), ou seja, igual a não utilização de qualquer filtro.

© 2000-2024, MetaQuotes Ltd.

FrameNext

Lê um quadro e move o ponteiro para o próximo. Existem duas variantes da função.

1. Chamando para receber um valor numérico

```
bool  FrameNext(
   ulong&   pass,       // O número de uma passagem na otimização, durante o qual o qua
   string&  name,       // Nome público/etiqueta
   long&    id,         // ID pública
   double&  value       // Valor
   );
```

2. Chamando para receber todos os dados de um quadro

```
bool  FrameNext(
   ulong&   pass,       // O número de uma passagem na otimização, durante o qual o qua
   string&  name,       // Nome público/etiqueta
   long&    id,         // ID pública
   double&  value,      // Valor
   void&    data[]      // Array de qualquer tipo
   );
```

Parâmetros

pass

[out] O número de um passe durante a otimização do testador de estratégia.

name

[out] O nome do identificador.

id

[out] O valor do identificador.

value

[out] Um valor numérico único.

data

[out] Uma array de qualquer tipo.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

Observação

Na segunda versão da chamada, você deve manipular corretamente os dados recebidos no array *data[]*.

© 2000-2024, MetaQuotes Ltd.

FrameInputs

Recebe parâmetros de entrada, no qual o quadro com o número especificado de passe é formado.

```
bool  FrameInputs(
   ulong    pass,                // O número de um passe na otimização
   string&  parameters[],        // Um array de strings na forma "parameterN=valueN"
   uint&    parameters_count     // O número total de parâmetros
   );
```

Parâmetros

pass

[in] O número de um passe durante a otimização do testador de estratégia.

parameters

[out] Um array de uma string com a descrição de nomes e valores de parâmetros

parameters_count

[out] O número de elementos no array *parameters[]*.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

Observação

Tendo obtido o número de strings *parameters_count* no array *parameters[]*, pode organizar um loop para percorrer todos os registros. Isso ajudará você a encontrar os valores dos parâmetros de entrada de um Expert Advisor para o número de passe especificado.

© 2000-2024, MetaQuotes Ltd.

FrameAdd

Adicionar um quadro com dados. Existem duas variantes da função.

1. Adicionar dados a partir de um arquivo

```
bool  FrameAdd(
   const string  name,         // Nome público/etiqueta
   long          id,           // ID público
   double        value,        // Valor
   const string  filename      // Nome de um arquivo de dados
   );
```

2. Adicionando dados a partir de um array de qualquer tipo

```
bool  FrameAdd(
   const string  name,         // Nome público/etiqueta
   long          id,           // ID público
   double        value,        // Valor
   const void&   data[]        // Array de qualquer tipo
   );
```

Parâmetros

name

[in] Etiqueta de um quadro Público. Ele pode ser usado para um filtro na função FrameFilter().

id

[in] Um identificador público do quadro. Ele pode ser usado para um filtro na função FrameFilter().

value

[in] Um valor numérico para escrever no quadro. Ele é utilizado para transmitir um passe único como resultado da função OnTester().

filename

[in] O nome do arquivo que contém os dados para adicionar ao quadro. O arquivo deve ser localizar na pasta MQL5/Files.

data

[in] Uma array de um tipo qualquer para escrever no quadro. Passado por referência.

Valor do Retorno

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

ParameterGetRange

Recebe os dados na faixa de valores e o passo de alteração para uma variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia. Existem 2 variantes da função.

1. Recebimento de dados do parâmetro de entrada para o tipo integer

```
bool  ParameterGetRange(
   const string  name,          // nome do parâmetro (variável de entrada)
   bool&         enable,        // parâmetro de otimização habilitado
   long&         value,         // o valor do parâmetro
   long&         start,         // valor inicial
   long&         step,          // alterar passo
   long&         stop           // valor final
   );
```

2. Recebendo dados para o parâmetro de entrada tipo real

```
bool  ParameterGetRange(
   const string  name,          // nome do parâmetro (variável de entrada)
   bool&         enable,        // parâmetro de otimização habilitado
   double&       value,         // valor do parâmetro
   double&       start,         // valor inicial
   double&       step,          // alterar passo
   double&       stop           // valor final
   );
```

Parâmetros

name

[in] variável de entrada ID. Essas variáveis são parâmetros externos de uma aplicação. Seus valores podem ser especificados quando do lançamento num gráfico ou durante um único teste.

enable

[out] Sinaliza que este parâmetro pode ser usado para enumerar os valores durante a otimização no Testador de Estratégia.

value

[out] Valor do parâmetro.

start

[out] Valor do parâmetro inicial durante a otimização.

step

[out] Parâmetro altera o passo ao enumerar os seus valores.

stop

[out] Valor do parâmetro final durante a otimização.

Valor de retorno

© 2000-2024, MetaQuotes Ltd.

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

A função pode ser chamado apenas pelos manipuladores OnTesterInit(), OnTesterPass() e OnTesterDeinit(). Foi introduzida para receber os valores dos parâmetros de entrada do Expert Advisor e faixas de variação durante a otimização no Testador de Estratégia.

Quando chamado em OnTesterInit(), os dados obtidos podem ser usados para redefinir as regras de enumeração de qualquer variável de entrada usando ParameterSetRange(). Portanto, novo Start, Stop e valores do Step podem ser definidos no parâmetro de entrada e podem até mesmo serem completamente excluídos a partir da otimização, independentemente das configurações do Testador de Estratégia. Isto permite-lhe gerenciar a área dos parâmetros de entrada durante a otimização, excluindo alguns parâmetros a partir da otimização de acordo com os valores dos parâmetros fundamentais do Expert Advisor.

Exemplo:

```
#property description "Expert Advisor para a demonstração da função ParameterGetRange
#property description "Deve ser lançado no modo de otimização do Testador de Estratégi
//--- parâmetros de entrada
input int                 Input1=1;
input double              Input2=2.0;
input bool                Input3=false;
input ENUM_DAY_OF_WEEK    Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Expert Advisor é projetado para operação somente no Testador de Estratégia
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Deve ser lançado no modo de otimização do Testador de Estratégia!");
      //--- finalizar a operação do Expert Advisor com antecedência e remover do gráf
      return(INIT_FAILED);
     }
//--- completada com sucesso a inicialização
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função TesterInit                                                |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- exemplo de parâmetros de entrada tipo long
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
```

© 2000-2024, MetaQuotes Ltd.

```
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("Primeiro parâmetro");
   PrintFormat("%s=%d  habilitado=%s  a partir de %d para %d com passo=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- exemplo de parâmetros de entrada tipo double
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("Segundo parâmetro");
   PrintFormat("%s=%G  habilitado=%s  a partir de %G para %G com passo=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- exemplo de parâmetros de entrada tipo bool
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Terceiro parâmetro");
   PrintFormat("%s=%s  habilitado=%s  a partir de %s para %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- exemplo de parâmetros de entrada tipo enumerador
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Quarto parâmetro");
   PrintFormat("%s=%s  habilitado=%s  a partir de %s para %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------+
//| Função TesterDeinit                                              |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- esta mensagem será mostrada após a otimização ser concluída
   Print(__FUNCTION__," Otimização concluída");
  }
```

© 2000-2024, MetaQuotes Ltd.

ParameterSetRange

Especifica o uso de variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia: valor, mudança de passo, os valores iniciais e finais. Existem 2 variantes da função.

1. Especificando os valores para o parâmetro de entrada do tipo integer

```
bool  ParameterSetRange(
   const string  name,          // nome do parâmetro (variável de entrada)
   bool          enable,        // parâmetro de otimização habilitado
   long          value,         // valor do parâmetro
   long          start,         // valor inicial
   long          step,          // mudar etapa
   long          stop           // valor final
   );
```

2. Especificando os valores para o parâmetro de entrada de tipo real

```
bool  ParameterSetRange(
   const string  name,          // nome do parâmetro (variável de entrada)
   bool          enable,        // parâmetro de otimização habilitado
   double        value,         // valor dos parâmetros
   double        start,         // valor inicial
   double        step,          // mudar etapa
   double        stop           // valor final
   );
```

Parâmetros

name

[in] input ou sinput ID variável. Essas variáveis são parâmetros externos de uma aplicação. Seus valores podem ser especificados quando iniciar o programa.

enable

[in] Ativar este parâmetro para enumerar os valores durante a otimização no Testador de Estratégia.

value

[in] valor do parâmetro.

start

[in] Valor do parâmetro inicial durante a otimização.

step

[in] Alteração de parâmetro etapa ao enumerar seus valores.

stop

[in] Valor do parâmetro final durante a otimização.

Valor de retorno

© 2000-2024, MetaQuotes Ltd.

Retorna true se bem sucedido, caso contrário false. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

A função só pode ser chamado pelo manipulador OnTesterInit() ao lançar otimização a partir do Testador de Estratégia. Ele é projetado para especificar o alcance do parâmetro e alterar o passo. O parâmetro pode ser completamente excluído da otimização, independentemente das configurações do Testador de Estratégia. Ele também permite o uso de variáveis declaradas com modificador de entrada no processo de otimização.

A função ParameterSetRange() permite que você gerencie uma otimização Expert Advisor no Testador de Estratégia dependendo dos valores de seus parâmetros fundamentais, incluindo ou excluindo os parâmetros de entrada necessários a partir da otimização e definir o intervalo necessário e o passo de mudança.

© 2000-2024, MetaQuotes Ltd.

Funções de Eventos

Este grupo contém funções para trabalhar com eventos personalizados e eventos de timer. Além deste grupo, há funções especiais para manipulação de eventos pré-definidos.

Função	Ação
EventSetMillisecondTimer	Inicia gerador de eventos de timer de alta resolução com um período de menos de 1 segundo para o gráfico em curso
EventSetTimer	Inicia o gerador de eventos do time com a periodicidade especificada para o gráfico atual
EventKillTimer	Interrompe a geração de eventos pelo timer no quadro atual
EventChartCustom	Gera um evento personalizado para o gráfico especificado

Também Veja

Tipos de Eventos de Gráficos

© 2000-2024, MetaQuotes Ltd.

EventSetMillisecondTimer

A função indica ao terminal de cliente que eventos de timer devem ser gerados em intervalos inferiores a um segundo, para este Expert Advisor ou aquele indicador.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // número de milissegundos
   );
```

Parâmetros

milliseconds

[in] Número de milissegundos que definem a freqüência de eventos de timer.

Valor de retorno

No caso de sucesso na execução, retorna true, do contrário - false. Para receber um código de erro, a função GetLastError() deverá ser utilizada.

Observação

Este recurso é projetado para os casos em que é necessário timer de alta resolução. Em outras palavras, eventos de timer devem ser recebidos com mais freqüência do que uma vez por segundo. Se um timer convencional, com o período de mais de um segundo é o suficiente para você, use o evento EventSetTimer().

Em geral, quando o período do timer é reduzido, o tempo de teste é maior, então o manipulador de eventos de timer é utilizado mais frequentemente. Ao trabalhar em modo de tempo real, os eventos de timer são gerados não mais do que uma vez em 10-16 milissegundos, devido a limitações de hardware.

Normalmente, esta função será chamada pela função OnInit() ou no construtor de classe. Para manipular eventos provenientes do timer, um Expert Advisor ou um indicador deve usar a função OnTimer().

Cada Expert Advisor e cada indicador trabalha com o seu próprio timer, recebendo eventos exclusivamente deste timer. Durante fechamento do aplicativo MQL5, o cronômetro é forçosamente encerrado caso tenha sido criado, mas não foi desativado pela função EventKillTimer().

Apenas um timer pode ser utilizado para cada programa. Cada aplicativo MQL5 e gráfico tem sua própria lista de eventos, onde todos os novos eventos são colocados. Se a lista já contém eventos de Timer ou este evento está em fase de processamento, então o novo evento de Timer não é adicionado a fila do aplicativo MQL5.

© 2000-2024, MetaQuotes Ltd.

EventSetTimer

A função indica ao terminal do cliente que para este indicador ou Expert Advisor, eventos do timer devem ser gerados com a periodicidade especificada.

```
bool  EventSetTimer(
   int  seconds      // número de segundos
   );
```

Parâmetros

seconds

[in] Número de segundos que determinam a freqüência de ocorrência do evento timer.

Valor do Retorno

Em caso de sucesso retorna verdadeiro, caso contrário falso. A fim de obter o código de erro, a função GetLastError() deve ser chamada.

Observação

Normalmente, esta função deve ser chamada a partir da função OnInit() ou de uma classe construtor. A fim de manipular eventos provenientes do timer, o Expert Advisor deve ter a função OnTimer().

Todo Expert Advisor, assim como todo indicador funciona com seu próprio timer e recebe eventos apenas a partir dele. Assim quando um programa MQL5 deixa de funcionar, o timer é destruído forçadamente se ele foi criado, mas ainda não foi desativado pela função EventKillTimer().

Para cada programa não mais de que um timer pode ser executado. Cada programa MQL5 e cada gráfico tem a sua própria fila de eventos em que todos os eventos recentemente recebidos são colocados. Se o evento Timer está presente ou está a ser processado, o novo evento de timer não será colocado na fila do programa MQL5.

© 2000-2024, MetaQuotes Ltd.

EventKillTimer

Especifica o terminal do cliente o que é necessário para finalizar a geração de eventos do Timer.

```
void  EventKillTimer();
```

Valor do Retorno

Sem valor de retorno.

Observação

Normalmente, esta função deve ser chamada a partir de uma função OnDeinit(), se a função EventSetTimer() for chamada de OnInit(). Esta função também pode ser chamada para formar o destruidor de classe, se a função EventSetTimer() foi chamada no construtor dessa classe.

Todo Expert Advisor, assim como todo indicador funciona com seu próprio timer e recebe eventos apenas a partir dele. Assim quando programa MQL5 deixa de funcionar, o timer é destruído forçadamente se ele foi criado, mas ainda não foi desativado pela função EventKillTimer().

© 2000-2024, MetaQuotes Ltd.

EventChartCustom

A função gera um evento personalizado para o gráfico especificado.

```
bool  EventChartCustom(
   long    chart_id,            // identificador do evento recebendo gráfico
   ushort  custom_event_id,     // identificador de evento
   long    lparam,              // parâmetro do tipo long
   double  dparam,              // parâmetro do tipo double
   string  sparam               // parâmetro de string do evento
   );
```

Parâmetros

chart_id

[in] Identificador gráfico. Significa o gráfico atual.

custom_event_id

[in] ID dos usuários do evento. Este identificador é automaticamente adicionado ao valor CHARTEVENT_CUSTOM e convertido para o tipo integer.

lparam

[in] Evento parâmetro do tipo long passado para a função OnChartEvent.

dparam

[in] Evento parâmetro do tipo double passado para a função OnChartEvent.

sparam

[in] Evento parâmetro do tipo string passado para a função OnChartEvent. Se a string é maior do que 63 caracteres, será truncado.

Valor do Retorno

Retorna verdadeiro se um evento personalizado foi colocada com sucesso na fila do gráfico de eventos que recebe os eventos. Em caso de falha, ele retorna false. Use GetLastError() para obter o código de erro.

Observação

Um Expert Advisor ou o indicador ligado a tabela especificada manipula o evento utilizando a função OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam).

Para cada tipo de evento, os parâmetros de entrada da função OnChartEvent() têm valores definidos que são requeridos para o processamento deste evento. Os eventos e valores passados através destes parâmetros são listados na tabela abaixo.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
Evento de uma teclada	CHARTEVENT_KEYDOWN	código de uma tecla pressionada	Repita a contagem (o número de vezes	O valor da string de uma pequena máscara de

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
			que a tecla é repetida como um resultado de que o usuário pressiona a tecla)	descrever o estado de botões do teclado
Evento do mouse (se a propriedade CHART_EVENT_MOUSE_MOVE=true, então é definido para o gráfico)	CHARTEVENT_MOUSE_MOVE	a coordenada X	a coordenada Y	O valor de string de uma máscara de bites descrevendo o estado de botões de mouse
Evento de criação de objeto gráfico (se CHART_EVENT_OBJECT_CREATE=true, então é definido para o gráfico)	CHARTEVENT_OBJECT_CREATE	—	—	Nome do objeto gráfico criado
Evento de mudança de uma propriedade de objeto via janela de diálogo de propriedades	CHARTEVENT_OBJECT_CHANGE	—	—	Nome do objeto gráfico modificado
Caso de apagamento objeto gráfico (se CHART_EVENT_OBJECT_DELETE=true, então é definido para o gráfico)	CHARTEVENT_OBJECT_DELETE	—	—	Nome do objeto gráfico excluído
Evento de um click de mouse no gráfico	CHARTEVENT_CLICK	a coordenada X	a coordenada Y	—
Evento de um click de mouse	CHARTEVENT_OBJECT_CLICK	a coordenada X	a coordenada Y	Nome do objeto gráfico, na qual

© 2000-2024, MetaQuotes Ltd.

Evento	Valor do parâmetro id	Valor do parâmetro lparam	Valor do parâmetro dparam	Valor do parâmetro sparam
num objeto gráfico pertencente ao gráfico				o evento ocorreu
Evento de um objeto gráfico arrastado usando o mouse	CHARTEVENT_OBJECT_DRAG	—	—	Nome do objeto gráfico movido
Evento da edição de texto finalizada na caixa de entrada do objeto gráfico LabelEdit	CHARTEVENT_OBJECT_ENDEDIT	—	—	Nome do objeto gráfico LabelEdit, cuja edição de texto foi concluída
Evento de mudança no gráfico	CHARTEVENT_CHART_CHANGE	—	—	—
ID do evento de usuário sob N número	CHARTEVENT_CUSTOM+N	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()	Valor definido pela função EventChartCustom()

Exemplo:

```
//+------------------------------------------------------------------+
//|                                              ButtonClickExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Botão";
string labelID="Informação";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| Função de inicialização do Expert                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Criar um botão para enviar eventos personalizados
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
```

```
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Botão");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- Criar uma etiqueta para exibir informações
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"Não informado");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Função de Desinicialização do Expert                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Função tick (ponto) de um Expert                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Verifique o evento pressionando um botão do mouse
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
```

```
      string clickedChartObject=sparam;
      //--- Se você clicar sobre o objeto com o nome buttonID
      if(clickedChartObject==buttonID)
        {
         //--- Estado do botão - pressionado ou não
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- registrar uma mensagem de depuração
         Print("Botão pressionado = ",selected);
         int customEventID; // Número do evento personalizado para enviar
         string message;    // Mensagem a ser enviada no caso
         //--- Se o botão for pressionado
         if(selected)
           {
            message="Botão pressionado";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // Botão não está pressionado
           {
            message="Botão não está pressionado";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- Enviar um evento personalizado "nosso" gráfico
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         ///--- Envie uma mensagem para todos os gráficos abertos
         BroadcastEvent(ChartID(),0,"Transmissão de mensagem");
         //--- Depurar mensagem
         Print("Enviar um evento com ID = ",customEventID);
        }
      ChartRedraw();// Redesenho forçado de todos os objetos de gráfico
     }

//--- Verificar o evento pertence ao evento de utilizador
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Tem transmissão de mensagem a partir de um gráfico com id = "+lparam);
        }
      else
        {
         //--- Nós lemos uma mensagem de texto no evento
         string info=sparam;
         Print("Manipular o evento do usuário com o ID = ",id);
         //--- Exibe uma mensagem numa etiqueta
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// Redesenho forçado de todos os objetos de gráfico
        }
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Enviar evento de transmissão para todos os gráficos abertos      |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                 // Temos, certamente, não mais do que CHARTS_MA
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // Recebemos um novo gráfico do anterior
      if(currChart==-1) break;        // Alcançado o final da lista de gráficos
      i++;// Não esqueça de aumentar o contador
     }
  }
//+------------------------------------------------------------------+
```

Também Veja

Eventos do Terminal Cliente, Funções de Manipulação de Evento

© 2000-2024, MetaQuotes Ltd.

Trabalhando com OpenCL

Programas em OpenCLsão utilizados para a realização de cálculos sobre placas de vídeo que suportam OpenCL 1.1 ou superior. Placas de vídeo modernas contêm centenas de pequenos processadores especializados que podem executar simultaneamente operações matemáticas simples, com fluxos de dados de entrada. A linguagem OpenCL organiza computação paralela e proporciona maior velocidade para uma certa classe de tarefas.

Em algumas placas de vídeo, o modo com números de tipo double está desativado por padrão, o que leva a o erro de compilação 5105. Para ativar o modo de suporte de números double, no texto do programa OpenCl, é preciso adicionar a diretiva #pragma OPENCL EXTENSION cl_khr_fp64 : enable. No entanto, se a placa de vídeo não suportar double, a desativação dessa diretiva não irá ajudar.

É recomendado escrever o código-fonte OpenCL em arquivos CL separados, que podem então ser ligados ao programa MQL5 usando variáveis de recursos.

Processamento de erros em programas OpenCL

Para obter informações sobre o erro mais recente no OpenCL, use as funções CLGetInfoInteger e CLGetInfoString, que proporcionam um código de erro e uma descrição de texto.

Código do último erro OpenCL: Para obter o erro OpenCL mais recente, use a chamada CLGetInfoInteger. Aqui o parâmetro *handle* é ignorado, ou seja, não importa (você pode especificar zero). Descrição de erros: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Para o código de erro desconhecido, "unknown OpenCL error N" é retornado, aqui N é o código de erro. Exemplo:

```
//--- o primeiro identificador de parâmetro é ignorado ao receber o código do último e
int code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Descrição de texto do erro OpenCL: Para obter o erro OpenCL mais recente, use a chamada CLGetInfoString. Aqui é necessário passa o código de erro como parâmetro *handle*.

Descrição de erros: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Se for passado CL_LAST_ERROR no lugar do código do erro, então a função retornará a descrição do último erro. Exemplo:

```
//--- obtém o código do último erro OpenCL
int    code = (int)CLGetInfoInteger(0,CL_LAST_ERROR);
string desc; // obtém a descrição de texto do erro

//--- usa o código de erro para obter uma descrição de texto do erro
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);

//--- para obter a descrição do último erro OpenCL sem obter o código primeiro, é nece
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION, desc))
   desc = "cannot get OpenCL error description, " + (string)GetLastError();
Print(desc);;
```

© 2000-2024, MetaQuotes Ltd.

Atualmente, como explicação do erro, é fornecido o nome da enumeração interna cuja decodificação pode ser encontrada na página https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Por exemplo, se o valor CL_INVALID_KERNEL_ARGS for recebido, a descrição dele será "Returned when enqueuing a kernel when some kernel arguments have not been set or are invalid."

Funções para os programas em execução em OpenCL:

Função	Ação
CLHandleType	Retorna o tipo do manipulador de OpenCL como valor da enumeração ENUM_OPENCL_HANDLE_TYPE
CLGetInfoInteger	Retorna o valor da propriedade de uma integer para um objeto ou dispositivo OpenCL
CLContextCreate	Cria um contexto OpenCL
CLContextFree	Remove um contexto OpenCL
CLGetDeviceInfo	Receives device property from OpenCL driver
CLProgramCreate	Cria um programa OpenCL a partir de um código fonte
CLProgramFree	Remove um programa OpenCL
CLKernelCreate	Cria uma função de partida OpenCL
CLKernelFree	Remove uma função de partida OpenCL
CLSetKernelArg	Define um parâmetro para a função OpenCL
CLSetKernelArgMem	Define um buffer OpenCL como um parâmetro da função OpenCL
CLSetKernelArgMemLocal	
CLBufferCreate	Cria um buffer OpenCL
CLBufferFree	Exclui um buffer OpenCL
CLBufferWrite	Grava um array num buffer OpenCL
CLBufferRead	Lê um buffer de OpenCL num array
CLExecute	Executa um programa OpenCL
CLExecutionStatus	

Também Veja

OpenCL, Recursos

© 2000-2024, MetaQuotes Ltd.

CLHandleType

Retorna o tipo de um manipulador de OpenCL como um valor do enumerador ENUM_OPENCL_HANDLE_TYPE.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // Manipulação de um objeto OpenCL
   );
```

Parâmetros

handle

[in] Um manipulador para um objeto OpenCL: um contexto, um kernel ou um programa OpenCL.

Valor do Retorno

O tipo de manipulador de OpenCL como valor do enumerador ENUM_OPENCL_HANDLE_TYPE.

ENUM_OPENCL_HANDLE_TYPE

Identificador	Descrição
OPENCL_INVALID	Manipulador incorreto
OPENCL_CONTEXT	Um manipulador de contexto OpenCL
OPENCL_PROGRAM	Um manipulador do programa OpenCL
OPENCL_KERNEL	Um manipulador do kernel OpenCL
OPENCL_BUFFER	Um manipulador do buffer OpenCL

© 2000-2024, MetaQuotes Ltd.

CLGetInfoInteger

Retorna o valor de uma propriedade integer para um objeto ou dispositivo OpenCL.

```
long  CLGetInfoInteger(
   int  handle,                          // O manipulador do objeto OpenCL ou o númer
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // Propriedade solicitada
   );
```

Parâmetros

handle

[in] Um manipulador para o objeto OpenCL ou o número do dispositivo OpenCL. A numeração dos dispositivos OpenCL começa com um zero.

prop

[in] O tipo de uma propriedade solicitada do enumerador ENUM_OPENCL_PROPERTY_INTEGER, o valor que você deseja obter.

Valor do Retorno

O valor da propriedade, se for bem-sucedido ou -1 em caso de erro. Para obter informações sobre o erro, utilize a função GetLastError().

ENUM_OPENCL_PROPERTY_INTEGER

Identificador	Descrição	Tipo
CL_DEVICE_COUNT	O número de dispositivos com suporte a OpenCL. Esta propriedade não requer a especificação do primeiro parâmetro, ou seja, você pode passar um valor zero para o parâmetro do *manipulador*.	int
CL_DEVICE_TYPE	Tipo de dispositivo	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	Identificador único do fornecedor	uint
CL_DEVICE_MAX_COMPUTE_UNITS	Número de tarefas paralelas calculados em dispositivos OpenCL. Um grupo de trabalho resolve uma tarefa computacional. O menor valor é 1	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	Define a maior freqüência do dispositivo em MHz.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	Tamanho da memória global do dispositivo em bites	ulong

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição	Tipo
CL_DEVICE_LOCAL_MEM_SIZE	Tamanho dos dados processados (cena) de memória local, em bites	uint
CL_BUFFER_SIZE	Actual size of the OpenCL buffer in bytes.	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	Número total de grupos locais de trabalho disponíveis para um dispositivo OpenCL.	ulong
CL_KERNEL_WORK_GROUP_SIZE	Número total de grupos locais de trabalho disponíveis para um programa OpenCL.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	Tamanho da memória local em bytes, que é usada pelo OpenCL do programa para todas as tarefas simultâneas no grupo. Utilize CL_DEVICE_LOCAL_MEM_SIZE para obter o máximo disponível.	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	Tamanho mínimo de memória privada em bytes a ser usado por cada tarefa na kernel do programa OpenCL.	ulong
CL_LAST_ERROR	Valor do último último erro OpenCL	int

O enumerador ENUM_CL_DEVICE_TYPE contém possíveis tipos de dispositivos que suportam OpenCL. Você pode receber o tipo de dispositivo pelo seu número ou o manipulador do objeto OpenCL, chamando CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE).

ENUM_CL_DEVICE_TYPE

Identificador	Descrição
CL_DEVICE_ACCELERATOR	Aceleradores OpenCL dedicados (por exemplo, a IBM CELL Blade). Estes dispositivos se comunicam com o processador hospedeiro utilizando uma interconexão de periféricos, como PCIe.
CL_DEVICE_CPU	Um dispositivo OpenCL que é o processador hospedeiro. O processador hospedeiro executa

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	as implementações OpenCL, sendo uma única ou multi-core CPU.
CL_DEVICE_GPU	Um dispositivo OpenCL que é uma GPU.
CL_DEVICE_DEFAULT	O dispositivo padrão OpenCL no sistema. O dispositivo padrão não pode ser um dispositivo CL_DEVICE_TYPE_CUSTOM.
CL_DEVICE_CUSTOM	Aceleradores dedicados que não suportam programas escritos em OpenCL C.
CL_ERROR_DESCRIPTION	Descrição de texto do erro OpenCL

Exemplo:

```
void OnStart()
  {
   int cl_ctx;
//--- inicializar contexto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL não encontrado");
      return;
     }
//--- Exibir informações sobre a plataforma
   Print("Tipo OpenCL: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("Fornecedor ID OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("Unidades OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("Freqüência OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY),
   Print("Memória global OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE),
   Print("Memória local OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE),"
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLGetInfoString

Retorna valor string de uma propriedade para o objeto OpenCL ou dispositivo.

```
bool  CLGetInfoString(
   int  handle,                          // Objeto manipulador de OpenCL ou número de
   ENUM_OPENCL_PROPERTY_STRING  prop,    // propriedade solicitada
   string&  value                        // string referenciada
   );
```

Parâmetros

handle

[in] Objeto manipulador de OpenCL ou número de dispositivo OpenCL. A numeração dos dispositivos OpenCL começa com um zero.

prop

[in] Tipo de propriedade requerida do enumerador ENUM_OPENCL_PROPERTY_STRING, o valor do qual deve ser obtido.

value

[out] String para receber o valor da propriedade.

Valor de retorno

true se bem sucedido, caso contrário false. Para obter informações sobre o erro, utilize a função GetLastError().

ENUM_OPENCL_PROPERTY_STRING

Identificador	Descrição
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - Perfil OpenCL. Nome do perfil pode ser um dos seguintes valores: • FULL_PROFILE - implementação suporta OpenCL (funcionalidade é definida como a parte da especificação do kernel sem necessidade de extensões adicionais para o apoio OpenCL); • EMBEDDED_PROFILE - implementação suporta OpenCL como um suplemento. Perfil alterado é definido como um subconjunto para cada versão OpenCL.
CL_PLATFORM_VERSION	Versão OpenCL
CL_PLATFORM_VENDOR	Nome do fornecedor da Plataforma
CL_PLATFORM_EXTENSIONS	Lista de extensões suportadas pela plataforma. Nomes de extensão deve ser suportado por todos os dispositivos relacionados com esta plataforma
CL_DEVICE_NAME	O nome do dispositivo

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
CL_DEVICE_VENDOR	Nome do fornecedor
CL_DRIVER_VERSION	Versão do driver OpenCL em formato major_number.minor_number
CL_DEVICE_PROFILE	Perfil de dispositivo OpenCL. Nome do perfil pode ser um dos seguintes valores: • FULL_PROFILE - implementação suporta OpenCL (funcionalidade é definida como a parte da especificação do kernel sem necessidade de extensões adicionais para o apoio OpenCL); • EMBEDDED_PROFILE - implementação suporta OpenCL como um suplemento. Perfil alterado é definido como um subconjunto para cada versão OpenCL.
CL_DEVICE_VERSION	OpenCL versão em formato "OpenCL<space><major_version.minor_version><space><vendor-specific information>"
CL_DEVICE_EXTENSIONS	Lista de extensões suportadas pelo dispositivo. A lista pode conter extensões suportadas pelo fornecedor, bem como um ou mais nomes aprovados: cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	A lista de kernels suportados separados por ";".
CL_DEVICE_OPENCL_C_VERSION	A versão máximo suportado pelo compilador para este dispositivo. Versão formato: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "

Exemplo:

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- inicializar contexto OpenCL
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL não encontrado");
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- Exibir informações sobre a plataforma
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("Nome da plataforma OpenCL: ");
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("Fornecedor da plataforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("Versão da plataforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("Perfil da plataforma OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("Extensão da plataforma OpenCL: ",str);
//--- Exibir informações sobre o dispositivo
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("Nome do dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("Perfil do dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("Kernels dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("Extensão do dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("Fornecedor do dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("Versão do dispositivo OpenCL: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("Versão do c OpenCL: ",str);
//--- Exibir informações gerais sobre o dispositivo OpenCL
   Print("Tipo OpenCL: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("Fornecedor ID OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("Unidades OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("Freqüência OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY))
   Print("Memória global OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE))
   Print("Memória local OpenCL: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLContextCreate

Cria um contexto OpenCL e retorna seu manipulador.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // Número de série do dispositivo OpenCL ou macro
   );
```

Parâmetro

device

[in] O número ordinal do dispositivo-OpenCL no sistema. Em vez de um número específico, você pode especificar um dos seguintes valores:

- CL_USE_ANY - Qualquer dispositivo disponível com suporte OpenCL é permitido;
- CL_USE_CPU_ONLY - Somente a emulação OpenCL na CPU é permitida;
- CL_USE_GPU_ONLY - A emulação OpenCL é proibida e somente dispositivos especializados com suporte OpenCL (placas de vídeo) podem ser usados;
- CL_USE_GPU_DOUBLE_ONLY - só GPUs que suportam o tipo podem ser usadas double.

Valor do Retorno

Um manipulador para o contexto OpenCL se for bem sucedido, caso contrário, -1. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLContextFree

Remove um contexto OpenCL.

```
void  CLContextFree(
   int  context     // Manipulador para um contexto OpenCL
   );
```

Parâmetros

context

[in] Manipula o contexto OpenCL.

Valor do Retorno

Nenhum. No caso de um erro interno o valor do modificador _LastError. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLGetDeviceInfo

A função recebe a propriedade do dispositivo do driver OpenCL.

```
bool  CLGetDeviceInfo(
   int     handle,          // manipulador de dispositivo OpenCL
   int     property_id,     // solicitado propriedade ID
   uchar&  data[],          // array para recebimento de dados
   uint&   size             // deslocamento na array de elementos, 0 por padrão
   );
```

Parâmetros

manipulador

[in] OpenCL identificador de dispositivo ou identificador OpenCL criado pela função CLContextCreate().

property_id

[in] ID da propriedade do dispositivo OpenCL que deve ser recebida. Os valores podem ser de um dos predeterminados listados na tabela abaixo.

data[]

[out] O array para receber dados sobre a propriedade solicitada.

tamanho

[out] Tamanho dos dados recebidos no array *data[]*.

Valor de retorno

true se bem sucedido, caso contrário false. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

Para array unidimensionais, o número do elemento, com o qual a leitura de dados para a escrita num buffer de OpenCL começa, é calculado tendo em conta os sinalizadores AS_SERIES.

A lista de IDs disponíveis das propriedades do dispositivo OpenCL

Descrição exata da propriedade e suas funções podem ser encontradas no site oficial do OpenCL.

Identificador	Valor
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

Exemplo:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERRO na CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("tamanho = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- Exemplo de entradas no diário Experts
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
```

© 2000-2024, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-37
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2024, MetaQuotes Ltd.

CLProgramCreate

Cria um programa OpenCL a partir de um código fonte.

```
int  CLProgramCreate(
   int           context,     // Manipulador para um contexto OpenCL
   const string  source       // O código-fonte
   );
```

Parâmetros

context

[in] Manipula o contexto OpenCL.

source

[in] String com o código-fonte do programa OpenCL.

Valor do Retorno

Um manipulador para um objeto OpenCL se for bem sucedido. Em caso de erro é retornado -1. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

Até o momento, são utilizados os seguintes códigos de erro:

- ERR_OPENCL_INVALID_HANDLE - manipulador inválido para lidar com o OpenCL contexto.
- ERR_INVALID_PARAMETER - parâmetro inválido de string.
- ERR_NOT_ENOUGH_MEMORY - não há memória suficiente para concluir a operação.
- ERR_OPENCL_PROGRAM_CREATE - erro interno do OpenCL ou erro de compilação.

Em alguns cartões gráficos, o trabalho com números do tipo double são desativados por padrão. Isso pode levar ao erro de compilação 5105. Para ativar o suporte para os números de tipo duplo, adicione a seguinte diretiva ao seu programa OpenCL: #pragma OPENCL EXTENSION cl_khr_fp64 : ativado

Exemplo:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
const string
cl_src=
       //--- por padrão alguns GPUs não suportam doubles
       //--- a directiva cl_khr_fp64 é usada para habilitar o trabalho com doubles
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- função OpenCL kernel
       "__kernel void Test_GPU(__global double *data,      \r\n"
       "                       const    int N,             \r\n"
       "                       const    int total_arrays)  \r\n"
       "  {                                                 \r\n"
       "   uint kernel_index=get_global_id(0);              \r\n"
       "   if (kernel_index>total_arrays) return;           \r\n"
```

© 2000-2024, MetaQuotes Ltd.

```
        "   uint local_start_offset=kernel_index*N;         \r\n"
        "   for(int i=0; i<N; i++)                          \r\n"
        "     {                                             \r\n"
        "       data[i+local_start_offset] *= 2.0;          \r\n"
        "     }                                             \r\n"
        "  }                                                \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- checar tamanho do array
   if(ArraySize(data)==0) return(false);
//--- checar índice do array
   if(id>total_arrays) return(false);
//--- Calcular deslocamento local para o array com id do índice
   int local_start_offset=id*N;
//--- multiplicar elementos por 2
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // tamanho do array
#define TOTAL_ARRAYS 5    // total de arrays
//--- manipulador OpenCL
int cl_ctx;  // contexto do manipulador OpenCL
int cl_prg;  // programa do manipulador OpenCL
int cl_krn;  // Kernel do manipulador OpenCL
int cl_mem;  // buffer do manipulador OpenCL
//---
double DataArray1[]; // dados do array para cálculo do CPU
double DataArray2[]; // dados do array para cálculo do GPU
//+------------------------------------------------------------------+
//| Função de início do programa script                              |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- inicializar objetos do OpenCL
//--- Criar contexto do OpenCL
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL não encontrado. Error=",GetLastError());
      return(1);
     }
//--- criar programa OpenCL
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       CLContextFree(cl_ctx);
       Print("criação programa OpenCL falhou. Error=",GetLastError());
       return(1);
      }
//--- criar kernel do OpenCL
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
      {
       CLProgramFree(cl_prg);
       CLContextFree(cl_ctx);
       Print("criação do kernel do OpenCL falhou. Error=",GetLastError());
       return(1);
      }
//--- criar buffer do OpenCL
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
      {
       CLKernelFree(cl_krn);
       CLProgramFree(cl_prg);
       CLContextFree(cl_ctx);
       Print("criação do buffer do OpenCL falhou. Error=",GetLastError());
       return(1);
      }
//--- Definir parâmetros de constante do kernel OpenCL
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- preparar dados dos arrays
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- preencher arrays com dados
   for(int j=0; j<TOTAL_ARRAYS; j++)
      {
       //--- Calcular o deslocamento de partida local para a jth array
       uint local_offset=j*ARRAY_SIZE;
       //--- preparar array com o índice j
       for(int i=0; i<ARRAY_SIZE; i++)
         {
          //--- Preencher arrays com funções MathCos(i+j);
          DataArray1[i+local_offset]=MathCos(i+j);
          DataArray2[i+local_offset]=MathCos(i+j);
         }
      };
//--- teste de cálculo do CPU
   for(int j=0; j<TOTAL_ARRAYS; j++)
      {
       //--- cálculo do array com o índice j
       Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
      }
//--- preparar os parâmetros do CLExecute
```

© 2000-2024, MetaQuotes Ltd.

```
   uint  offset[]={0};
//--- tamanho do trabalho global
   uint  work[]={TOTAL_ARRAYS};
//--- escrever dados para o buffer do OpenCL
   CLBufferWrite(cl_mem,DataArray2);
//--- executar kernel do OpenCL
   CLExecute(cl_krn,1,offset,work);
//--- Ler dados do buffer OpenCL
   CLBufferRead(cl_mem,DataArray2);
//--- total de erros
   double total_error=0;
//--- Comparar os resultados e calcular o erro
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- Calcular deslocamento local para o jth array
      uint local_offset=j*ARRAY_SIZE;
      //--- comparar os resultados
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- Mostrar primeiro e último arrays
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- deletar objetos do OpenCL
//--- buffer OpenCL livre
   CLBufferFree(cl_mem);
//--- kernel OpenCL livre
   CLKernelFree(cl_krn);
//--- programa OpenCL livre
   CLProgramFree(cl_prg);
//--- contexto OpenCL livre
   CLContextFree(cl_ctx);
//---
   return(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

CLProgramFree

Remove um programa OpenCL.

```
void  CLProgramFree(
   int  program     // Manipulador para um objeto OpenCL
   );
```

Parâmetros

program

[in] Manipulador para um objeto OpenCL.

Valor do Retorno

Nenhum. No caso de um erro interno o valor do modificador _LastError. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLKernelCreate

Cria o kernel do programa OpenCL e retorna seu manipulador.

```
int  CLKernelCreate(
   int           program,        // Manipulador para um objeto OpenCL
   const string  kernel_name     // Nome do kernel
   );
```

Parâmetros

program

[in] Manipulador para um objeto do programa OpenCL.

kernel_name

[in] O nome da função do kernel no programa apropriado OpenCL, em que a execução se inicia.

Valor do Retorno

Um manipulador para um objeto OpenCL se for bem sucedido. Em caso de erro é retornado -1. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

Até o momento, são utilizados os seguintes códigos de erro:

- ERR_OPENCL_INVALID_HANDLE - manipulador inválido para lidar com o *programa* OpenCL.
- ERR_INVALID_PARAMETER - parâmetro inválido de string.
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - nome do kernel contém mais de 127 caracteres.
- ERR_OPENCL_KERNEL_CREATE - Ocorreu um erro interno durante a criação de um objeto OpenCL.

© 2000-2024, MetaQuotes Ltd.

CLKernelFree

Remove uma função inicial OpenCL.

```
void  CLKernelFree(
   int  kernel     // Manipulador para o kernel de um programa OpenCL
   );
```

Parâmetros

kernel_name

[in] Manipulador do objeto do kernel.

Valor do Retorno

Nenhum. No caso de um erro interno o valor do modificador _LastError. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArg

Define um parâmetro para a função OpenCL.

```
bool  CLSetKernelArg(
   int   kernel,        // Manipulador para o kernel de um programa OpenCL
   uint  arg_index,     // O número do argumento da função OpenCL
   void  arg_value      // Código-fonte
   );
```

Parâmetros

kernel

[in] Manipulador para um kernel do programa OpenCL.

arg_index

[in] O número do argumento da função, a numeração começa com zero.

arg_value

[in] O valor do argumento da função.

Valor do Retorno

Retorna true se for bem sucedido, caso contrário retorna false. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

Até o momento, são utilizados os seguintes códigos de erro:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE – manipulador inválido para o kernel OpenCL.
- ERR_OPENCL_SET_KERNEL_PARAMETER - erro interno do OpenCL.

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArgMem

Define um buffer OpenCL como um parâmetro da função OpenCL.

```
bool  CLSetKernelArgMem(
   int   kernel,           // Manipulador para o kernel de um programa OpenCL
   uint  arg_index,        // O número do argumento da função OpenCL
   int   cl_mem_handle     // Manipulador para buffer de OpenCL
   );
```

Parâmetros

kernel

[in] Manipulador para um kernel do programa OpenCL.

arg_index

[in] O número do argumento da função, a numeração começa com zero.

cl_mem_handle

[in] Um manipulador para um buffer OpenCL.

Valor do Retorno

Retorna true se for bem sucedido, caso contrário retorna false. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLSetKernelArgMemLocal

Ele define o buffer local como argumento da função kernel.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // identificador para a kernel do programa OpenCL
   uint   arg_index,        // número do argumento OpenCL de função
   ulong  local_mem_size    // tamanho do buffer
   );
```

Parâmetros

kernel

[in] Identificador para a kernel do programa OpenCL.

arg_index

[in] Número de argumento de função, a numeração começa em zero.

local_mem_size

[in] Tamanho do buffer em bytes.

Valor de retorno

Retorna true se a execução for bem-sucedida, caso contrario, false. Para obter informações sobre o erro utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLBufferCreate

Cria um buffer OpenCL e retorna seu manipulador.

```
int  CLBufferCreate(
   int   context,      // Manipulador para um contexto OpenCL
   uint  size,         // Tamanho do buffer
   uint  flags         // Combinação de sinalizadores que especifique as propriedades d
   );
```

Parâmetros

context

[in] Um manipulador de contexto OpenCL.

size

[in] Tamanho do buffer em bites.

flags

[in] Propriedades de buffer que são definidas usando uma combinação de sinalizadores: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

Valor do Retorno

Um manipulador para um buffer OpenCL se for bem sucedido. Em caso de erro é retornado -1. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

Até o momento, são utilizados os seguintes códigos de erro:

- ERR_OPENCL_INVALID_HANDLE - inválido manipulador ao contexto OpenCL.
- ERR_NOT_ENOUGH_MEMORY – memória insuficiente.
- ERR_OPENCL_BUFFER_CREATE – erro interno criando buffers.

© 2000-2024, MetaQuotes Ltd.

CLBufferFree

Exclui um buffer de OpenCL.

```
void  CLBufferFree(
   int   buffer     // Manipulador para um buffer OpenCL
   );
```

Parâmetros

buffer

[in] Um manipulador para um buffer OpenCL.

Valor do Retorno

Nenhum. No caso de um erro interno o valor do modificador _LastError. Para obter informações sobre o erro, utilize a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

CLBufferWrite

Escreve para o buffer OpenCL e retorna o número de elementos escritos.

```
uint  CLBufferWrite(
   int          buffer,                    // Um manipulador para o buffer OpenCL
   const void&  data[],                    // Uma array de valores
   uint         buffer_offset=0,           // Um deslocamento no buffer OpenCL em bite
   uint         data_offset=0,             // Um deslocamento na array de elementos, (
   uint         data_count=WHOLE_ARRAY     // O número de valores a partir do array pa
   );
```

Existem também versões para trabalhar com matrizes e vetores.

Escreve os valores do array para o buffer e retorna true se bem-sucedido.

```
uint  CLBufferWrite(
   int           buffer,                   // Um manipulador para o buffer OpenCL
   uint          buffer_offset,            // O deslocamento no buffer OpenCL em byte
   matrix<T>     &mat                      // O array de valores para registro no bu
   );
```

Escreve valores para o buffer a partir do vetor e retorna true se bem sucedido.

```
uint  CLBufferWrite(
   int           buffer,                   // Um manipulador para o buffer OpenCL
   uint          buffer_offset,            // O deslocamento no buffer OpenCL em byte
   vector<T>     &vec                      // O vetor de valores a serem escritos no
   );
```

Parâmetros

buffer

[in] O manipulador do buffer OpenCL.

data[]

[in] Um array de valores que deve ser escrito no buffer OpenCL. Passado por referência.

buffer_offset

[in] Uma compensação no buffer OpenCL em bites, a partir do qual começa a escrita. Por padrão, a escrita começa com o início do buffer.

data_offset

[in] O índice do primeiro elemento array, começando a partir do qual valores de array são escritos no buffer OpenCL. Por padrão, são tomados os valores desde o início do array.

data_count

[in] O número de valores que devem ser escritos. Todos os valores do array, por padrão.

mat

[out] O array para a leitura dos dados do buffer pode ser de qualquer um dos três tipos — matrix, matrixf ou matrixc.

© 2000-2024, MetaQuotes Ltd.

vec

[out] O vetor de leitura dos dados do buffer pode ser de qualquer um dos três tipos — vector, vectorf ou vectorc.

Valor do Retorno

O número de elementos escritos. 0 (zero) é devolvido em caso de erro. Para obter informações sobre o erro, utilize a função GetLastError().

true quando executado com sucesso ao trabalhar com uma matriz ou vetor, no caso de um erro, ele retorna false.

Observação

Para arrays unidimensionais, o número do elemento, com o qual a leitura de dados para a escrita num buffer de OpenCL começa, é calculado tendo em conta os sinalizadores AS_SERIES.

Um array de duas ou mais dimensões é apresentado como unidimensional. Neste caso, *data_offset* é o número de elementos que devem ser ignorados na apresentação, e não o número de elementos na primeira dimensão.

Exemplo de multiplicação de arrray por meio do método MatMul e computação paralela em OpenCL

```
#define M       3000      // O número de filas no primeiro array
#define K       2000      // O número de colunas no primeiro array é igual ao número d
#define N       3000      // O número de colunas no segundo array

//+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                 \r\n"
  "__kernel void matricesMul( __global float *in1,                  \r\n"
  "                           __global float *in2,                  \r\n"
  "                           __global float *out  )                \r\n"
  "{                                                                \r\n"
  "  int m = get_global_id( 0 );                                    \r\n"
  "  int n = get_global_id( 1 );                                    \r\n"
  "  float sum = 0.0;                                               \r\n"
  "  for( int k = 0; k < K; k ++ )                                  \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                 \r\n"
  "  out[ m * N + n ] = sum;                                        \r\n"
  "}                                                                \r\n";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- Inicializa o gerador de números aleatórios
  MathSrand((int)TimeCurrent());
```

```
//--- preenche o array de um determinado tamanho com valores aleatórios
  matrixf mat1(M, K, MatrixRandom) ;     // O primeiro array
  matrixf mat2(K, N, MatrixRandom);      // O segundo arraty

//--- Calcula o produto dos arrays de uma maneira ingênua
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// aqui escrevemos o resultado da multipli
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- Calcula o produto dos arrays por meio de MatMull
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- Calculamos o produto de dois arrays em OpenCL
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // O manipulador do contexto
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL não encontrado, saímos");
    return;
   }
  int cl_prg;              // O manipulador do programa
  int cl_krn;              // O manipulador do kernel
  int cl_mem_in1;          // O manipulador do primeiro buffer (de saída)
  int cl_mem_in2;          // O manipulador do segundo buffer (de saída)
  int cl_mem_out;          // O manipulador do terceiro buffer (de saída)
//--- Cria o programa e o kernel
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- Cria os três buffers para os três arrays
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//--- terceiro array - de saída
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- Define o argumento do kernel
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- Escreve os arrays nos buffers do dispositivo
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- Início do tempo de execução do código OpenCL
  start=GetTickCount();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- Define os parâmetros da área de trabalho e executa o programa OpenCL
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- Calcula o resultado no array
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("Matriz lida [%d x %d]: ", matrix_opencl.Rows(), matrix_opencl.Cols())
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error ",GetLastError())
  uint time_opencl=GetTickCount()-start;
  Print("Comparamos o tempo de cálculo com cada método");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- Libera todos os contextos OpenCL
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- Compara todos os arrays de resultados obtidos uns com os outros
  Print("Quantos erros de divergência existem entre os arrays de resultados?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  Resultado:

   Matriz lida [3000 x 3000]:
  Compara o tempo de cálculo com cada método
   Naive product time = 54750 ms
   MatMul product time = 4578 ms
   OpenCl product time = 922 ms
   Quantos erros de divergência existem entre os arrays de resultados?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Preenche o array com valores aleatórios                          |
//+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
   }
```

```
 }
//+------------------------------------------------------------------+
//| Libera todos os contextos OpenCL                                 |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- Remove qualquer contexto OpenCL criado em ordem inversa
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

© 2000-2024, MetaQuotes Ltd.

CLBufferRead

Lê um buffer de OpenCL em uma array e retorna o número de elementos de leitura.

```
uint  CLBufferRead(
   int          buffer,                    // Um manipulador para o buffer OpenCL
   const void&  data[],                    // Uma array de valores
   uint         buffer_offset=0,           // Um deslocamento no buffer OpenCL em bite
   uint         data_offset=0,             // Um deslocamento na array de elementos, (
   uint         data_count=WHOLE_ARRAY     // O número de valores a partir da buffer c
   );
```

Existem também versões para trabalhar com matrizes e vetores.

Lê o buffer OpenCL em um array e retorna true se for bem-sucedido.

```
uint  CLBufferRead(
   int           buffer,                   // Um manipulador para o buffer OpenCL
   uint          buffer_offset,            // O deslocamento no buffer OpenCL em byte
   const matrxi& mat,                      // Um array para obter o valor desde o bu
   ulong         rows=-1,                  // A quantidade de linhas no array
   ulong         cols=-1                   // A quantidade de colunas no array
   );
```

Lê o buffer OpenCL no vetor e retorna true se bem-sucedido.

```
uint  CLBufferRead(
   int           buffer,                   // Um manipulador para o buffer OpenCL
   uint          buffer_offset,            // O deslocamento no buffer OpenCL em byte
   const vector& vec,                      // O vetor para obter valores a partir do
   ulong         size-1,                   // O tamanho do vetor
   );
```

Parâmetros

buffer

[in] Um manipulador do buffer OpenCL.

data[]

[in] Um array para receber os valores do buffer OpenCL. Passado por referência.

buffer_offset

[in] Um deslocamento no buffer OpenCL em bites, a partir da qual se inicia a leitura. Por padrão, a leitura começa com o início do buffer.

data_offset

[in] O índice do primeiro elemento array para escrever os valores de buffer OpenCL. Por padrão, a escrita dos valores lidos em um array começa a partir do índice zero.

data_count

[in] O número de valores que devem ser lidos. Todo o buffer OpenCL é lido por padrão.

mat

© 2000-2024, MetaQuotes Ltd.

[out] O array para a leitura dos dados do buffer pode ser de qualquer um dos três tipos — matrix, matrixf ou matrixc.

vec

[out] O vetor de leitura dos dados do buffer pode ser de qualquer um dos três tipos — vector, vectorf ou vectorc.

rows=-1

[in] Se o parâmetro for especificado, o parâmetro *cols* também deve ser especificado. Se não forem especificadas novas dimensões do array, serão utilizadas as dimensões atuais. Se o valor for -1, o número de linhas não mudará.

cols=-1

[in] Se não for especificado nenhum parâmetro, o parâmetro *rows* também deve ser omitido. A regra básica para um array é especificar ambos os parâmetros, ou nenhum deles, caso contrário, ocorrerá um erro. Se os dois parâmetros (*rows* e *cols*) forem especificados, a matriz será redimensionada. Se o valor for -1, o número de colunas não mudará.

size=-1

[in] Se o parâmetro não for especificado ou o valor for -1, o comprimento do vetor não mudará.

Valor do Retorno

O número de elementos de leitura. 0 (zero) é devolvido em caso de erro. Para obter informações sobre o erro, utilize a função GetLastError().

true quando executado com sucesso ao trabalhar com uma matriz ou vetor, no caso de um erro, ele retorna false.

Observação

Para arrays unidimensionais, o número do elemento, em que a escrita de dados num buffer de OpenCL inicia, é calculado tendo em conta os sinalizadores AS_SERIES.

Um array de duas ou mais dimensões é apresentado como unidimensional. Neste caso, *data_offset* é o número de elementos que devem ser ignorados na apresentação, e não o número de elementos na primeira dimensão.

Exemplode cálculo do número Pi com a fórmula:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
#define  _intrnCnt         _num_steps / _divisor

//+------------------------------------------------------------------+
```

```
//|                                                                  |
//+------------------------------------------------------------------+
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
string I2S(int arg)                { return IntegerToString(arg); }

//--- OpenCL programm code
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi Calculation: step = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- prepare OpenCL contexts
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);

  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- launch OpenCL program
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
   {
    Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
```

```
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- get results from OpenCL device
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
   {
    Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
    return(-1);
   }
//--- sum all values to calculate Pi
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculated for "+D2S(time, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- free memory
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- success
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Auxiliary routine to free memory                                 |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

© 2000-2024, MetaQuotes Ltd.

CLExecute

A função executa um programa OpenCL. Existem três versões da função:

1. Lançamento das funções kernel usando um kernel

```
bool  CLExecute(
   int          kernel                    // Manipulador para o kernel de um programa
   );
```

2. Lançamento de várias cópias do kernel (função OpenCL) com descrição do espaço de tarefas

```
bool  CLExecute(
   int          kernel,                   // Manipulador para o kernel de um programa
   uint         work_dim,                 // Dimensão do espaço de tarefas
   const uint&  global_work_offset[],     // Compensação inicial no espaço de tarefas
   const uint&  global_work_size[]        // Número total de tarefas
   );
```

3. Lançamento de várias cópias do kernel (função OpenCL) com descrição do espaço de tarefa e especificação do tamanho de subconjunto de tarefa local do grupo

```
bool  CLExecute(
   int          kernel,                   // Manipulador para o kernel de um programa
   uint         work_dim,                 // Dimensão do espaço de tarefas
   const uint&  global_work_offset[],     // Compensação inicial no espaço de tarefas
   const uint&  global_work_size[],       // Número total de tarefas
   const uint&  local_work_size[]         // Número de tarefas no grupo local
   );
```

Parâmetros

kernel

[in] Manipulador para o kernel OpenCL.

work_dim

[in] Dimensão do espaço tarefas.

global_work_offset[]

[in] Compensação inicial no espaço de tarefas.

global_work_size[]

[in] O tamanho de um subconjunto de tarefas.

local_work_size[]

[in] O tamanho do subconjunto de tarefa local do grupo.

Valor do Retorno

Retorna true se for bem sucedido, caso contrário retorna false. Para obter informações sobre o erro, utilize a função GetLastError().

Observação

© 2000-2024, MetaQuotes Ltd.

Considerar o uso dos parâmetros no exemplo a seguir:

- *work_dim* especifica *work_items[]* da dimensão do array descrevendo as tarefas. Se *work_dim*=3, array tridimensional *work_items[N1, N2, N3]* é usado.
- *global_work_size[]* contém os valores definindo o tamanho array do *work_items[]*. Se work_dim=3, array do *global_work_size[3]* pode ser {40, 100, 320}. Então nós temos *work_items[40, 100, 320]*. Assim, o número total de tarefas é 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* define o subconjunto das tarefas que serão executadas pelo kernel específico de programa OpenCL. A sua dimensão é igual a dimensão*work_items[]* e permite a divisão exata da tarefa comum de subconjunto dentro de subconjuntos menores. Na verdade, os tamanhos do array do *local_work_size[]* devem ser selecionados de modo a dividir a tarefa global *work_items[]* definindo dentro de subconjuntos menores. *local_work_size[3]={10, 10, 10}* vai se encaixar no exemplo corrente, como *work_items[40, 100, 320]* podem ser obtidos a partir do array do *local_items[10, 10, 10]* sem excesso.

© 2000-2024, MetaQuotes Ltd.

CLExecutionStatus

Retornar o estado de execução do programa OpenCL.

```
int  CLExecutionStatus(
   int   kernel            // identificador para a kernel do programa OpenCL
   );
```

Parâmetros

kernel

[in] Identificador para a kernel do programa OpenCL.

Valor de retorno

Retorna o estado do programa OpenCL, o valor pode ser um dos seguintes:

- CL_COMPLETE=0 - programa concluído,
- CL_RUNNING=1 – em execução,
- CL_SUBMITTED=2 – enviado para execução,
- CL_QUEUED=3 – encontra-se na fila para execução,
- -1 (menos um) – teve lugar erro durante a execução de CLExecutionStatus().

© 2000-2024, MetaQuotes Ltd.

As funções de banco de dados usam o SQLite, um mecanismo popular e fácil de usar. Esse mecanismo é conveniente porque todo o banco de dados fica em um único arquivo no computador do usuário.

Usando essas funções, é fácil criar tabelas, adicionar dados a elas, fazer modificações e fazer seleções com consultas SQL simples:

- receber histórico de negociação e cotações de qualquer formato,
- salvar resultados de otimização e de teste,
- preparar e intercambiar dados com outros pacotes de análise,
- salvar configurações e status de programas MQL5.

Além disso, você pode usar em consultas funções estatísticas e matemáticas.

As funções para trabalhar com bancos de dados permitem substituir as operações mais repetitivas, quanto ao processamento de grandes matrizes de dados, por consultas SQL, o que permite, em muitos casos, usar chamadas em vez de programar loops e comparações complexas DatabaseExecute/DatabasePrepare. Para obter convenientemente os resultados de consulta em uma estrutura pronta, use a função DatabaseReadBind, que permite ler todos os campos de registro em uma chamada diretamente.

Para acelerar as operações de leitura, gravação e modificação, o banco de dados pode ser aberto/criado no banco de dados online com o sinalizador DATABASE_OPEN_MEMORY, sendo que esse banco de dados estará disponível apenas para esse programa específico e não será compartilhado com ninguém. Ao trabalhar com bancos de dados localizados no disco rígido, é necessário embrulhar grandes inserções/alterações de dados em uma transação usando DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback, o que fornece aceleração centenas de vezes.

Para começar a trabalhar com estas funções, basta ler o artigo SQLite: trabalho nativo com bancos de dados SQL em MQL5.

Função	Ação
DatabaseOpen	Abre ou cria um banco de dados no arquivo especificado
DatabaseClose	Fecha um banco de dados
DatabaseImport	Importa para uma tabela dados de um arquivo
DatabaseExport	Exporta uma tabela ou resultado sobre a execução da consulta SQL para um arquivo CSV
DatabasePrint	Exibe uma tabela ou resultado sobre a execução da consulta SQL no log do EA
DatabaseTableExists	Verifica a existência de tabela no banco de dados
DatabaseExecute	Executa uma consulta no banco de dados especificado
DatabasePrepare	Cria um identificador de consulta, que pode ser executado usando DatabaseRead()
DatabaseReset	Redefine a consulta para seu status inicial, como após uma chamada de DatabasePrepare()

© 2000-2024, MetaQuotes Ltd.

Função	Ação
DatabaseBind	Define o valor do parâmetro na consulta
DatabaseBindArray	Define a matriz como um valor de parâmetro
DatabaseRead	Realiza o a transição para o próximo registro como resultado de uma consulta
DatabaseReadBind	Moves to the next record and reads data into the structure from it
DatabaseFinalize	Exclui uma consulta criada em DatabasePrepare()
DatabaseTransactionBegin	Inicia a execução da transação
DatabaseTransactionCommit	Conclui a execução da transação
DatabaseTransactionRollback	Executa a reversão de transações
DatabaseColumnsCount	Obtém o número de campos na consulta
DatabaseColumnName	Obtém o nome do campo pelo número
DatabaseColumnType	Obter tipo de campo por número
DatabaseColumnSize	Obtém o tamanho do campo em bytes
DatabaseColumnText	Obtém o valor do campo do registro atual como uma string
DatabaseColumnInteger	Obtém um valor int do registro atual
DatabaseColumnLong	Obtém um valor long do registro atual
DatabaseColumnDouble	Obtém um valor double do registro atual
DatabaseColumnBlob	Obtém o valor do campo do registro atual como um array

Funções estatísticas:

- mode — moda
- median — mediana (50° percentil)
- percentile_25 — 25° percentil
- percentile_75 — 75° percentil
- percentile_90 — 90° percentil
- percentile_95 — 95° percentil
- percentile_99 — 99° percentil
- stddev или stddev_samp — desvio padrão da amostra
- stddev_pop — desvio padrão da população
- variance or var_samp — variância da amostra
- var_pop — variação da população

Funções matemáticas

- acos(X) - arco cosseno em radianos

© 2000-2024, MetaQuotes Ltd.

- acosh(X) - arco cosseno inverso hiperbólico
- asin(X) - arco seno em radianos
- asinh(X) - arco seno hiperbólico
- atan(X) - arco tangente em radianos
- atan2(X,Y) - arco tangente em radianos da relação X/Y
- atanh(X) - arco tangente hiperbólico
- ceil(X) - arredondando para um número inteiro
- ceiling(X) - arredondamento para um número inteiro
- cos(X) - cosseno de um ângulo em radianos
- cosh(X) - cosseno hiperbólico
- degrees(X) - converte radianos em ângulo
- exp(X) - exponente
- floor(X) - arredondamento para um número inteiro
- ln(X) - logaritmo natural
- log(B,X) - logaritmo para a base indicada
- log(X) - logaritmo decimal
- log10(X) - logaritmo decimal
- log2(X) - logaritmo de base 2
- mod(X,Y) - restante da divisãopi() - número aproximado de Pi
- pow(X,Y) - potência de uma determinada base
- power(X,Y) – potência de uma determinada base
- radians(X) - converte o ângulo em radianos
- sin(X) - seno do ângulo em radianos
- sinh(X) - seno hiperbólico
- sqrt(X) - raiz quadrada
- tan(X) - tangente de um ângulo em radianos
- tanh(X) - tangente hiperbólica
- trunc(X) - reduz para o número inteiro mais próximo de 0

Exemplo:

```
select
  count(*) as book_count,
  cast(avg(parent) as integer) as mean,
  cast(median(parent) as integer) as median,
  mode(parent) as mode,
  percentile_90(parent) as p90,
  percentile_95(parent) as p95,
  percentile_99(parent) as p99
from moz_bookmarks;
```

© 2000-2024, MetaQuotes Ltd.

DatabaseOpen

Abre ou cria um banco de dados no arquivo especificado.

```
int  DatabaseOpen(
   string  filename,       // nome do arquivo
   uint    flags,          // combinação de sinalizadores
   );
```

Parâmetros

filename

[in] Nome do arquivo relativo à pasta "MQL5\Files".

flags

[in] Combinação de sinalizadores da enumeração ENUM_DATABASE_OPEN_FLAGS.

Valor retornado

Se for bem-sucedida, a função retorna o identificador do banco de dados, que é usado para acessar os dados do banco de dados, caso contrário, retorna INVALID_HANDLE. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - erro interno ao acessar a pasta "MQL5\Files";
- ERR_INVALID_PARAMETER (4003) - caminho para o arquivo de banco de dados contendo uma string vazia ou definida uma combinação incompatível de sinalizadores;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_FILENAME (5002) - nome de arquivo de banco de dados inválido;
- ERR_TOO_LONG_FILENAME (5003) - caminho absoluto para o arquivo de banco de dados excedendo o tamanho máximo;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - excedido número máximo de objetos Database;
- ERR_DATABASE_CONNECT (5123) - erro de conexão ao banco de dados;
- ERR_DATABASE_MISUSE (5621) - uso incorreto da biblioteca SQLite.

Geral

Se no parâmetro *filename* estiver especificado NULL ou uma string vazia "", no disco será criado um arquivo temporário, que será excluído automaticamente após encerrada a conexão com o banco de dados.

Se no parâmetro *filename* estiver especificadoн ":memory:", o banco de dados será criado na memória e o banco de dados será excluído automaticamente após encerrada a conexão.

Se no parâmetro *flags* não estiver especificado nenhum do sinalizadores DATABASE_OPEN_READONLY ou DATABASE_OPEN_READWRITE, será usado o sinalizador DATABASE_OPEN_READWRITE.

Se não estiver especificada a extensão do arquivo, será usada a extensão ".sqlite"

ENUM_DATABASE_OPEN_FLAGS

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
DATABASE_OPEN_READONLY	Abrir somente leitura
DATABASE_OPEN_READWRITE	Abrir para leitura e escrita
DATABASE_OPEN_CREATE	Criar um arquivo no disco se ele não existir
DATABASE_OPEN_MEMORY	Criar um banco de dados na RAM
DATABASE_OPEN_COMMON	O arquivo está localizado na pasta compartilhada dos terminais

Veja também

DatabaseClose

© 2000-2024, MetaQuotes Ltd.

DatabaseClose

Закрывает базу данных.

```
void  DatabaseClose(
   int  database      // identificador de banco de dados recebido em DatabaseOpen
   );
```

Parâmetros

database

[in] Identificador de banco de dados recebido em DatabaseOpen().

Valor retornado

Não.

Observação

Após chamar DatabaseClose, todos os identificadores de consulta de banco de dados são excluídos automaticamente e se tornam inválidos.

No caso de um identificador válido, a função mostrará o erro ERR_DATABASE_INVALID_HANDLE. O erro pode ser verificado usando GetLastError().

Veja também

DatabaseOpen, DatabasePrepare

© 2000-2024, MetaQuotes Ltd.

DatabaseImport

Importa para uma tabela dados de um arquivo.

```
long  DatabaseImport(
   int           database,          // identificador de banco de dados recebido em Dat
   const string  table,             // nome da tabela para inserir dados
   const string  filename,          // nome do arquivo para importar dados
   uint          flags,             // combinação de sinalizadores
   const string  separator,         // separador de dados
   ulong         skip_rows,         // quantas primeiras linhas ignorar
   const string  skip_comments      // strings de símbolos que definem comentários
   );
```

Parâmetros

database

[in] Identificador do banco de dados obtido em DatabaseOpen().

table

[in] Nome da tabela à qual serão adicionados os dados do arquivo.

filename

[in] Arquivo CSV ou arquivo ZIP para leitura de dados, o nome pode conter subpastas e é definido em relação à pasta MQL5\Files.

flags

[in] Combinação de sinalizadores a partir da enumeração ENUM_DATABASE_IMPORT_FLAGS.

separator

[in] Separador de dados em um arquivo CSV.

skip_rows

[in] Número de linhas iniciais a serem ignoradas ao ler dados de um arquivo.

skip_comments

[in] Linhas dos símbolos para designar strings como comentários. Se, no início da linha, for encontrado qualquer símbolo de *skip_comments*, tal linha é considerada um comentário e não é importada.

Valor retornado

Retorna o número de linhas importadas ou -1 em caso de erro. Para obter o código de erro, use GetLastError(), respostas possíveis:

- ERR_INVALID_PARAMETER (4003) - o nome da tabela não está especificado (linha vazia ou NULL);
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido.

Observação

© 2000-2024, MetaQuotes Ltd.

Se tabelas cujo nome é *table* não existirem, elas serão criadas automaticamente. Os nomes e o tipo de campos na tabela criada serão reconhecidos automaticamente com base nos dados contidos no arquivo.

ENUM_DATABASE_IMPORT_FLAGS

Identificador	Descrição
DATABASE_IMPORT_HEADER	A primeira linha contém os nomes dos campos da tabela
DATABASE_IMPORT_CRLF	Quebra de linha é CRLF (por padrão, LF)
DATABASE_IMPORT_APPEND	Adiciona dados ao final de uma tabela existente
DATABASE_IMPORT_QUOTED_STRINGS	Os valores de string estão contidos em aspas duplas
DATABASE_IMPORT_COMMON_FOLDER	O arquivo está localizado na pasta comum de todos os terminais do cliente \Terminal\Common\File.

Exemplo de leitura de tabela a partir de um arquivo criado pelo código no exemplo em DatabaseExport :

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- obtemos os nomes dos arquivos de texto a serem carregados da pasta comum dos te
  string filenames[];
  if(FileSelectDialog("Selecione o arquivo CSV para carregamento da tabela", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
    //--- exibimos o nome de cada arquivo selecionado
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Erro desconhecido durante a seleção do arquivo. Código de erro ", GetLas
      return;
     }
   }
  else
   {
    Print("arquivo CSV não selecionado");
    return;
```

© 2000-2024, MetaQuotes Ltd.

```
    }
//--- criamos ou abrimos um banco de dados
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- verificamos a presença da tabela TEST
  if(DatabaseTableExists(db, "TEST"))
   {
    //--- removemos a tabela TEST
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
     {
      Print("Failed to drop the TEST table with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   }
  //--- importamos os registros desde um arquivo para uma tabela TEST
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|DATABA
  if(imported>0)
   {
    Print(imported," lines imported in table TEST");
    DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
   }
  else
    {
     Print("DatabaseImport() failed. Error ",GetLastError());
    }
//--- fechamos o arquivo do banco de dados e relatamos isso
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

Veja também

DatabaseOpen, DatabasePrint

© 2000-2024, MetaQuotes Ltd.

DatabaseExport

Exporta uma tabela ou resultado sobre a execução da consulta SQL para um arquivo CSV. O arquivo é criado com codificação UTF-8.

```
long  DatabaseExport(
   int           database,          // identificador de banco de dados recebido em Da
   const string  table_or_sql,      // nome da tabela ou consulta SQL
   const string  filename,          // nome do arquivo CSV para exportação de dados
   uint          flags,             // combinação de sinalizadores
   const string  separator          // separador de dados no arquivo CVS
   );
```

Parâmetros

database

[in] Identificador do banco de dados, obtido em DatabaseOpen().

table_or_sql

[in] Nome da tabela ou o texto da consulta SQL, cujos resultados serão exportados para o arquivo especificado.

filename

[in] Nome do arquivo para exportar dados. O caminho é definido em relação à pasta MQL5\Files.

flags

[in] Combinação de sinalizadores a partir da enumeração ENUM_DATABASE_EXPORT_FLAGS.

separator

[in] Separador de dados. Se especificado NULL, como separador será utilizado o símbolo de tabulação '\t'. Linha vazia "" é considerada um delimitador válido, mas o arquivo CSV não pode ser lido como tabela, pois será um conjunto de linhas.

Valor retornado

Retorna o número de registros exportados ou um valor negativo em caso de erro. Para obter o código de erro, use GetLastError(), respostas possíveis:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_INVALID_PARAMETER (4003) - caminho para o arquivo de banco de dados contendo uma string vazia ou definida uma combinação incompatível de sinalizadores;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_FUNCTION_NOT_ALLOWED(4014) - o canal especificado não é permitido;
- ERR_PROGRAM_STOPPED(4022) - operação cancelada (o programa MQL parou);
- ERR_WRONG_FILENAME (5002) - nome de arquivo inválido;
- ERR_TOO_LONG_FILENAME (5003) - o caminho absoluto para o arquivo excedeu o tamanho máximo;
- ERR_CANNOT_OPEN_FILE(5004) - erro ao abrir o arquivo para gravação;
- ERR_FILE_WRITEERROR(5026) - erro ao gravar no arquivo;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;

© 2000-2024, MetaQuotes Ltd.

- ERR_DATABASE_QUERY_PREPARE(5125) - erro ao criar a consulta;
- ERR_DATABASE_QUERY_NOT_READONLY - somente é permitida uma solicitação de leitura.

Observação

Se os resultados da consulta forem exportados, a consulta SQL deverá começar com "SELECT" ou "select". Em outras palavras, a consulta SQL não pode alterar o estado do banco de dados, caso contrário DatabaseExport() falhará.

Os valores de string no banco de dados podem conter um caractere de conversão ('\r' ou '\r\n'), além de um caractere separador de valores especificado no parâmetro *separator*. Nesse caso, é necessário usar o sinalizador DATABASE_EXPORT_QUOTED_STRINGS no parâmetro flags. Se esse sinalizador estiver presente, todas as linhas exibidas serão colocadas entre aspas duplas, mas se a linha contiver aspas duplas, ela será substituída por duas aspas duplas.

ENUM_DATABASE_EXPORT_FLAGS

Identificador	Descrição
DATABASE_EXPORT_HEADER	Exibe os nomes dos campos na primeira linha
DATABASE_EXPORT_INDEX	Exibe os números das linhas
DATABASE_EXPORT_NO_BOM	Não insere uma etiqueta BOM no início do arquivo (a BOM é inserida por padrão)
DATABASE_EXPORT_CRLF	Usa CRLF (por padrão, LF) para quebras de linha
DATABASE_EXPORT_APPEND	Adiciona dados ao final de um arquivo existente (por padrão, o arquivo é sobregravado). Se o arquivo não existir, ele será criado.
DATABASE_EXPORT_QUOTED_STRINGS	Exibe valores de string em aspas duplas.
DATABASE_EXPORT_COMMON_FOLDER	O arquivo CSV será criado na pasta compartilhada de todos os terminais \Terminal\Common\File.

Exemplo:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
//--- lembramos a hora de início antes de obter as barras
  ulong start=GetMicrosecondCount();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- solicitamos as últimas 100 barras no período gráfico H1
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
   {
    Print("CopyRates() failed,, Error ", GetLastError());
    return;
   }
  else
   {
    //--- quantidade de barras e em quanto tempo elas foram recebidas
    PrintFormat("%s: CopyRates received %d bars in %d ms ",
                _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- formamos o nome do arquivo para armazenar o banco de dados
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())-
  StringReplace(filename, ":", "-"); // o símbolo ":" é proibido nos nomes dos arquivo
//--- abrimos/criamos um banco de dados na pasta comum do terminal
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|DATABASE_
  if(db==INVALID_HANDLE)
   {
    Print("Database: ", filename, " open failed with code ", GetLastError());
    return;
   }
  else
    Print("Database: ", filename, " opened successfully");

//--- verificamos a presença da tabela RATES
  if(DatabaseTableExists(db, "RATES"))
   {
    //--- removemos a tabela RATES
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
     {
      Print("Failed to drop the RATES table with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   }
//--- criamos a tabela RATES
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                      "SYMBOL             CHAR(10),"
                      "TIME               INT NOT NULL,"
                      "OPEN               REAL,"
                      "HIGH               REAL,"
                      "LOW                REAL,"
                      "CLOSE              REAL,"
                      "TICK_VOLUME        INT,"
                      "SPREAD             INT,"
                      "REAL_VOLUME        INT);"))
   {
    Print("DB: ", filename, " create table RATES with code ", GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
    DatabaseClose(db);
    return;
   }
//--- mostramos a lista de todos os campos na tabela RATES
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
   {
    PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d at
    DatabaseClose(db);
    return;
   }
//--- criamos uma consulta parametrizada para adicionar barras à tabela RATES
  string sql="INSERT INTO RATES (SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,RE
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // parâmetros da consulta
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
   {
    PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
    Print("SQL request: ", sql);
    DatabaseClose(db);
    return;
   }
//--- definimos o valor do primeiro parâmetro de consulta
  DatabaseBind(request, 0, _Symbol);
//--- lembramos a hora de início antes de adicionar as barras à tabela RATES
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
   {
    //--- definimos o resto dos parâmetros antes de adicionar uma entrada
    ResetLastError();
    if(!DatabaseBind(request, 1, rates[i].time))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    //--- se a chamada anterior de DatabaseBind() foi bem sucedida, definimos o seguin
    if(!request_error && !DatabaseBind(request, 2, rates[i].open))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 3, rates[i].high))
     {
```

```
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 4, rates[i].low))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 5, rates[i].close))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }

    //--- executamos uma solicitação para inserir um registro e verificar a existência
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
     {
      PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    //--- reinicializamos a solicitação para o estado inicial antes da próxima atualiz
    if(!request_error && !DatabaseReset(request))
     {
      PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
      DatabaseFinalize(request);
      request_error=true;
      break;
     }
   } //--- terminamos, percorremos todas as barras

//--- como correram as operações?
  if(request_error)
   {
    PrintFormat("Table RATES: failed to add %d bars ", ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Table RATES: added %d bars in %d ms",
                ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- salvamos a tabela RATES em um arquivo CSV
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_F
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- fechamos o arquivo do banco de dados e relatamos isso
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
```

Veja também

DatabasePrint, DatabaseImport

© 2000-2024, MetaQuotes Ltd.

DatabasePrint

Exibe uma tabela ou resultado sobre a execução da consulta SQL no log do EA.

```
long  DatabasePrint(
   int           database,          // identificador de banco de dados recebido em Dat
   const string  table_or_sql,      // tabela ou consulta SQL
   uint          flags              // combinação de sinalizadores
   );
```

Parâmetros

database

[in] Identificador do banco de dados, obtido em DatabaseOpen().

table_or_sql

[in] Nome da tabela ou o texto da consulta SQL, cujos resultados são exibidos no log do EA.

flags

[in] Combinação de sinalizadores que definem a formatação de exibição. Os sinalizadores são definidos da seguinte maneira:

DATABASE_PRINT_NO_HEADER - não exibir nomes de colunas da tabela (nomes de campos)
DATABASE_PRINT_NO_INDEX - não exibir os números de linha
DATABASE_PRINT_NO_FRAME - não exibir o quadro que separa o cabeçalho e os dados
DATABASE_PRINT_STRINGS_RIGHT - alinhar as linhas à direita.

Se flags=0, serão exibidas colunas e linhas, o cabeçalho e os dados são separados por um quadro, as linhas são alinhadas à esquerda.

Valor retornado

Retorna o número de linhas exportadas ou -1 em caso de erro. Para obter o código de erro, use GetLastError(), respostas possíveis:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;

Observação

Se os resultados da consulta forem exportados para o log, a consulta SQL deverá começar com "SELECT" ou "select". Em outras palavras, a consulta SQL não pode alterar o estado do banco de dados, caso contrário DatabasePrint() falhará.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="departments.sqlite";
```

© 2000-2024, MetaQuotes Ltd.

```
//--- criamos ou abrimos um banco de dados na pasta compartilhada do terminal
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- criamos a tabela COMPANY
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- criamos a tabela DEPARTMENT
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- exibimos a lista de todos os campos na tabela COMPANY e DEPARTMENT
   PrintFormat("Try to print request \"PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DE
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") failed, error code=%d", GetL
      DatabaseClose(db);
      return;
     }
//--- exibimos no log a tabela COMPANY
   PrintFormat("Try to print request \"SELECT * from COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- texto da consulta para junção das tabelas COMPANY e DEPARTMENT
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- exibimos o resultado da junção de tabelas
   PrintFormat("Try to print request \"SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUT
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- fechamos o banco de dados
```

© 2000-2024, MetaQuotes Ltd.

```
   DatabaseClose(db);
  }
/*
Conclusão:
Try to print request "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)"
#| cid name    type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+-----------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Try to print request "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+--------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMEN
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| Cria a tabela COMPANY                                            |
//+------------------------------------------------------------------+
bool CreataTableCompany(int database)
  {
//--- se a tabela COMPANY existir, vamos exclui-la
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- excluímos a tabela
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
        {
```

© 2000-2024, MetaQuotes Ltd.

```
         Print("Failed to drop table COMPANY with code ", GetLastError());
         return(false);
        }
     }
//--- criamos a tabela COMPANY
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: create table COMPANY failed with code ", GetLastError());
      return(false);
     }

//--- inserimos os dados na tabela COMPANY
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VAI
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'D
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'K
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
     {
      Print("COMPANY insert failed with code ", GetLastError());
      return(false);
     }
//--- sucesso
   return(true);
  }
//+------------------------------------------------------------------+
//| Cria a tabela DEPARTMENT                                         |
//+------------------------------------------------------------------+
bool CreataTableDepartment(int database)
  {
//--- se a tabela DEPARTMENT existir, vamos exclui-la
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- excluímos a tabela
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("Failed to drop table DEPARTMENT  with code ", GetLastError());
         return(false);
        }
     }
//--- criamos a tabela DEPARTMENT
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
                       "ID      INT PRIMARY KEY   NOT NULL,"
```

```
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
     {
      Print("DB: create table DEPARTMENT failed with code ", GetLastError());
      return(false);
     }

//--- inserimos os dados na tabela DEPARTMENT
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, '
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerin
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance',
     {
      Print("DEPARTMENT insert failed with code ", GetLastError());
      return(false);
     }
//--- sucesso
   return(true);
  }
//+------------------------------------------------------------------
```

Veja também

DatabaseExport, DatabaseImport

© 2000-2024, MetaQuotes Ltd.

DatabaseTableExists

Verifica a existência de tabela no banco de dados.

```
bool  DatabaseTableExists(
   int     database,      // identificador de banco de dados recebido em DatabaseOpen
   string  table          // nome da tabela
   );
```

Parâmetros

database

[in] Identificador do banco de dados, obtido em DatabaseOpen().

table

[in] Nome da tabela.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - nome da tabela não está especificado (string vazia ou NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta;
- ERR_DATABASE_NO_MORE_DATA (5126) - a tabela não existe (não é um erro, finalização normal);

Veja também

DatabasePrepare, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseExecute

Исполняет запрос к указанной базе данных.

```
bool  DatabaseExecute(
   int     database,      // identificador de banco de dados recebido em DatabaseOpen
   string  sql,           // consulta SQL
   );
```

Parâmetros

database

[in] Identificador do banco de dados, obtido em DatabaseOpen().

sql

[in] consulta SQL.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_INVALID_PARAMETER (4003) - parâmetro sql contendo string vazia;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta.

Example:

```
//--- symbol statistics
struct Symbol_Stats
  {
   string            name;             // symbol name
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and co
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swa
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
```

© 2000-2024, MetaQuotes Ltd.

```
  };

//--- Magic Number statistics
struct Magic_Stats
  {
   long              magic;            // EA's Magic Number
   int               trades;           // number of trades for the symbol
   double            gross_profit;     // total profit for the symbol
   double            gross_loss;       // total loss for the symbol
   double            total_commission; // total commission for the symbol
   double            total_swap;       // total swaps for the symbol
   double            total_profit;     // total profit excluding swaps and commissions
   double            net_profit;       // net profit taking into account swaps and com
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

//--- entry hour statistics
struct Hour_Stats
  {
   char              hour_in;          // market entry hour
   int               trades;           // number of trades in this entry hour
   double            volume;           // volume of trades in this entry hour
   double            gross_profit;     // total profit in this entry hour
   double            gross_loss;       // total loss in this entry hour
   double            net_profit;       // net profit taking into account swaps and com
   int               win_trades;       // number of profitable trades
   int               loss_trades;      // number of losing trades
   double            expected_payoff;  // expected payoff for the trade excluding swap
   double            win_percent;      // percentage of winning trades
   double            loss_percent;     // percentage of losing trades
   double            average_profit;   // average profit
   double            average_loss;     // average loss
   double            profit_factor;    // profit factor
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
```

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("Deals in the trading history: %d ", ExtDealsTotal);

//--- get trading statistics per symbols
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gro
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as trades
                               "   sum(case when profit > 0 then profit else 0 end) as
                               "   sum(case when profit < 0 then profit else 0 end) as
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_
                               "   sum(case when profit < 0 then 1 else 0 end) as loss
                               "   FROM DEALS "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY SYMBOL"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
```

```
//--- get records from request results
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("Trade statistics by Symbol");
   ArrayPrint(stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- get trading statistics for Expert Advisors by Magic Numbers
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trac
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT MAGIC,"
                           "   sum(case when entry =1 then 1 else 0 end) as trades,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(swap) as total_swap,"
                           "   sum(commission) as total_commission,"
                           "   sum(profit) as total_profit,"
                           "   sum(profit+swap+commission) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trac
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
```

© 2000-2024, MetaQuotes Ltd.

```
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("Trade statistics by Magic Number");
   ArrayPrint(EA_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accour
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- deals cannot be transformed to trades using a simple method through transa
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
```

```
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,S
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.hour as hour_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the table TRADES failed with code ", GetLastError());
         return;
        }

//--- get trading statistics by market entry hours
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT HOUR_IN,"
                           "   count() as trades,"
                           "   sum(volume) as volume,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(profit) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM TRADES "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                           "   GROUP BY HOUR_IN"
```

© 2000-2024, MetaQuotes Ltd.

```
                              "    ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- print
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- delete the request
   DatabaseFinalize(request);

//--- close database
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tot
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
```

© 2000-2024, MetaQuotes Ltd.

```
[6] "GBPUSD"      121     1668.50000  -1438.20000          -7.96000    -49.93000
[7] "USDCAD"       99      405.28000   -475.47000          -8.68000    -31.68000
[8] "USDCHF"      206     1588.32000  -1241.83000         -17.98000    -65.92000
[9] "USDJPY"      107      464.73000   -730.64000         -35.12000    -34.24000

Trade statistics by Magic Number
    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tota
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000

Trade statistics by entry hour
     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit] [win_trades]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000           21
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000           12
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000            5
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000           22
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000           27
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000           20
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000           11
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000            9
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000           24
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000           72
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000          115
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000           84
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000           84
[13]        13       64  6.40000      778.27000   -516.22000    262.05000           36
[14]        14       62  6.20000      536.93000   -427.47000    109.46000           38
[15]        15       50  5.00000      699.92000   -413.00000    286.92000           28
[16]        16       88  8.80000      778.55000   -514.00000    264.55000           51
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000           44
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000           24
[19]        19       52  5.20000      407.67000   -150.36000    257.31000           30
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000            9
[21]        21       10  1.00000       41.86000    -32.38000      9.48000            7
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000            6
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000            1
*/

//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
```

```
      {
       return(false);
      }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: create the DEALS table  failed with code ", GetLastError());
         return(false);
        }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- add deals to the table
   if(!InsertDeals(database))
      return(false);
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
```

© 2000-2024, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
   char     hour;                // deal execution hour
   MqlDateTime time_strusture;
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- add each deal to the table using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

© 2000-2024, MetaQuotes Ltd.

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Veja também

DatabasePrepare, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabasePrepare

Cria um identificador de consulta, que pode ser executado usando DatabaseRead().

```
int  DatabasePrepare(
   int     database,       // identificador de banco de dados recebido em DatabaseOpen
   string  sql,            // consulta SQL
           ...             // parâmetros de consulta
   );
```

Parâmetros

database

[in] Identificador do banco de dados, obtido em DatabaseOpen().

sql

[in] Consulta SQL que pode conter parâmetros de substituição automática com nomes ?1,?2,...

...

[in] Parâmetros de consulta substituídos automaticamente.

Valor retornado

Se for bem-sucedido, retorna o identificador para a consulta SQL, caso contrário, retorna INVALID_HANDLE. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - caminho para o arquivo de banco de dados contendo uma string vazia ou definida uma combinação incompatível de sinalizadores;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - excedido número máximo de objetos Database;
- ERR_DATABASE_PREPARE (5125) - erro ao criar consulta.

Observação

A função DatabasePrepare() não realiza a consulta relacionada ao banco de dados. Seu objetivo é verificar se os parâmetros da consulta são corretos e retornar o identificador com base nos resultados dessa verificação para executar a consulta SQL. A consulta em si será executada na primeira chamada de DatabaseRead().

Example:

```
//--- Structure to store the deal
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
   char              entry;            // DEAL_ENTRY
   string            symbol;           // DEAL_SYMBOL
```

© 2000-2024, MetaQuotes Ltd.

```
   double            volume;           // DEAL_VOLUME
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- Structure to store the trade: the order of members corresponds to the position i
struct Trade
  {
   datetime          time_in;          // entry time
   ulong             ticket;           // position ID
   char              type;             // buy or sell
   double            volume;           // volume
   string            symbol;           // symbol
   double            price_in;         // entry price
   datetime          time_out;         // exit time
   double            price_out;        // exit price
   double            commission;       // entry and exit commission
   double            swap;             // swap
   double            profit;           // profit or loss
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
```

© 2000-2024, MetaQuotes Ltd.

```
   PrintFormat("Deals in the trading history: %d ", deals_total);
//--- add deals to the table
   if(!InsertDeals(db))
      return;
//--- show the first 10 deals
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- print the deals
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("The first %d deals:", i);
      ArrayPrint(deals);
     }

//--- delete request after use
   DatabaseFinalize(request);

//--- make sure that hedging system for open position management is used on the accour
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- deals cannot be transformed to trades using a simple method through transa
```

© 2000-2024, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }

//--- now create the TRADES table based on the DEALS table
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- fill in the TRADES table using an SQL query based on DEALS table data
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- populate the TRADES table
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PR
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: fillng the TRADES table failed with code ", GetLastError());
         return;
        }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- show the first 10 deals
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   for(i=0; DatabaseReadBind(request, trade); i++)
     {
      if(i>=10)
```

© 2000-2024, MetaQuotes Ltd.

```
         break;
      trades[i].time_in=trade.time_in;
      trades[i].ticket=trade.ticket;
      trades[i].type=trade.type;
      trades[i].volume=trade.volume;
      trades[i].symbol=trade.symbol;
      trades[i].price_in=trade.price_in;
      trades[i].time_out=trade.time_out;
      trades[i].price_out=trade.price_out;
      trades[i].commission=trade.commission;
      trades[i].swap=trade.swap;
      trades[i].profit=trade.profit;
     }
//--- print trades
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nThe first %d trades:", i);
      ArrayPrint(trades);
      PrintFormat("Filling the TRADES table took %.2f milliseconds",double(transactior
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry] [s
   [0] 34429573              0                 0 2019.09.05 22:39:59      2       0 ""
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0       0 "U
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1       0 "U
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0       0 "E
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00      1       0 "A
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00      1       0 "U
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00      0       0 "U
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00      1       0 "E
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00      0       0 "C
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1       0 "E

   The first 10 trades:
                  [time_in] [ticket] [type] [volume] [symbol] [price_in]          [time
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06 18:
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06 18:
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09 03:
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09 03:
```

© 2000-2024, MetaQuotes Ltd.

```
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06 18:
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06 14:
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06 14:
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06 14:
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06 14:
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create the DEALS table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
```

```
      {
       Print("Failed to drop the DEALS table  with code ", GetLastError());
       return(false);
      }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number (Expert Advisor ID)
   long     reason;              // deal execution reason or source
//--- go through all deals and add them to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- lock the database before executing transactions
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal to the table using the following request
```

© 2000-2024, MetaQuotes Ltd.

```
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIN
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%d with code %d", __FUNCTION__, deal_
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- check for transaction execution errors
   if(failed)
     {
      //--- roll back all transactions and unlock the database
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code %d", __FUNCTION__, GetLastEr
      return(false);
     }
//--- all transactions have been performed successfully - record changes and unlock th
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the TRADES table                                         |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- if the TRADES table already exists, delete it
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- check if the table exists
   if(!DatabaseTableExists(database, "TRADES"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: create the TRADES table failed with code ", GetLastError());
         return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
```

Veja também

DatabaseExecute, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseReset

Redefine a consulta para seu status inicial, como após uma chamada de DatabasePrepare().

```
int  DatabaseReset(
   int  request      // identificador de consulta recebido em DatabasePrepare
   );
```

Parâmetros

request

[in] Identificador de consulta recebido em DatabasePrepare().

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- código de erro SQLite, começando com ERR_DATABASE_ERROR(5601).

Observação

A função DatabaseReset() é projetada para execução múltipla de uma consulta com diferentes valores de parâmetro. Por exemplo, ao adicionar em massa dados a uma tabela usando o comando INSERT, para cada registro, é necessário criar um conjunto próprio de valores para cada campo.

Ao contrário de DatabasePrepare(), a chamada de DatabaseReset() não compila strings com comandos SQL em uma nova consulta, por isso, DatabaseReset() é muito mais rápido do que DatabasePrepare().

DatabaseReset() é usado juntamente com a função DatabaseBind() e/ou DatabaseBindArray(), se for necessário alterar os valores dos parâmetros da consulta após a execução de DatabaseRead(). Isso significa que antes de definir novos valores de parâmetros de consulta (antes do bloco de chamada de DatabaseBind/DatabaseBindArray) é necessário chamar DatabaseReset() para redefini-lo ao seu estado inicial. A própria consulta parametrizada deve ser criada com a ajuda de DatabasePrepare().

Tanto DatabaseReset() quanto DatabasePrepare() não consultam o banco de dados. A consulta direta é executada ao chamar DatabaseRead() ou DatabaseReadBind().

A chamada de DatabaseReset() não redefine os valores dos parâmetros na consulta, se eles foram definidos através da chamada de DatabaseBind()/DatabaseBindArray(), ou seja, os parâmetros mantêm os seus valores. Assim, é possível alterar o valor de apenas um parâmetro, pois não há necessidade de registrar novamente todos os parâmetros de consulta após chamar DatabaseReset().

Não se pode passar ao DatabaseReset() o identificador da consulta que foi excluída com a ajuda de DatabaseFinalize(). Isso causa um erro.

Exemplo:

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

© 2000-2024, MetaQuotes Ltd.

```
//--- criamos ou abrimos o banco de dados
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- se a tabela SYMBOLS existir, vamos exclui-la
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- excluímos a tabela
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Failed to drop table SYMBOLS with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- criamos a tabela SYMBOLS
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- exibimos uma lista contendo todos os campos na tabela SYMBOLS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=%d
      DatabaseClose(db);
      return;
     }

//--- criamos a consulta parametrizada para adicionar símbolos à tabela SYMBOLS
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // parâmetros de consulta
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- percorremos todos os símbolos e os adicionamos à tabela SYMBOLS
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- definimos o valor dos parâmetros antes de adicionar um símbolo
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      //--- se a chamada anterior de DatabaseBind() foi bem-sucedida, definimos o segu
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
```

© 2000-2024, MetaQuotes Ltd.

```
          request_error=true;
          break;
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
          PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
          request_error=true;
          break;
        }

      //--- executamos a consulta para inserir o registro e verificamos se há erros
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
          PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
          break;
        }
      else
          PrintFormat("%d: added %s", i+1, symbol);
      //--- redefinimos a consulta para seu status inicial antes da seguinte atualizac
      if(!DatabaseReset(request))
        {
          PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
          DatabaseFinalize(request);
          request_error=true;
          break;
        }
    } //--- acabamos por aqui, passamos por todos os símbolos

//--- resultado da transação
   if(request_error)
     {
      PrintFormat("Table SYMBOLS: failed to add %d symbols", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS: added %d symbols",symbols);
     }

//--- salvamos a tabela SYMBOLS em um arquivo CSV
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("Database: table SYMBOLS saved in ", csv_filename);
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") failed with code", Ge

//--- fechamos o arquivo contendo o banco de dados e relatamos isso
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
//+------------------------------------------------------------------+
//| Retorna a especificação do símbolo como JSON                     |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- recuos
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)+"\","-
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol, SYMBOL_CURR
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)SymbolInfoIn
               "\n"+indent3+"\"Digits\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol, SYM
               "\n"+indent3+"\"TickBookDepth\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ChartMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeCalcMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"OrderMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"CalculationMode\":\""+IntegerToString(SymbolInfoIntegen
               "\n"+indent3+"\"ExecutionMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ExpirationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"FillFlags\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"ExpirFlags\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"Spread\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"TickValue\":\""+StringFormat("%G", (SymbolInfoDouble(sy
               "\n"+indent3+"\"TickSize\":\""+StringFormat("%G", (SymbolInfoDouble(syn
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(syn
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(syn
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
               "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
               "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
```

© 2000-2024, MetaQuotes Ltd.

```
                "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
                "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(sym
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Cria o recuo de espaços                                          |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 Resultado:
  Database: symbols.sqlite opened successfully
  #| cid name        type notnull dflt_value pk
  -+-------------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
  1: added EURUSD
  2: added GBPUSD
  3: added USDCHF
  ...
  82: added USDCOP
  83: added USDARS
  84: added USDCLP
  Table SYMBOLS: added 84 symbols
  Database: table SYMBOLS saved in symbols.csv
  Database: symbols.sqlite created and closed
*/
```

Veja também

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

© 2000-2024, MetaQuotes Ltd.

DatabaseBind

Define o valor do parâmetro na consulta.

```
bool  DatabaseBind(
   int  request,       // manipulador da consulta criada em DatabasePrepare
   int  index,         // índice do parâmetro na consulta
   T    value          // valor de parâmetro de tipo simples
   );
```

Parâmetros

request

[in] Identificador da consulta criada em DatabasePrepare().

index

[in] Índice do parâmetro na consulta para o qual é necessário definir o valor. A numeração começa do zero.

value

[in] Valor do parâmetro que é necessário definir. São permitidos os tipos: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - tipo não suportado;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_NOT_READY (5128) - não se pode usar a função para consulta no momento. A consulta está sendo realizada ou já foi concluída, é necessário chamar DatabaseReset().

Observação

Esta função deve ser usada caso a consulta SQL contenha os valores parametrizáveis "?" ou"?N", onde N é o número do parâmetro (começando de um). Ao mesmo tempo, a indexação de parâmetros no DatabaseBind() começa do zero.

Por exemplo:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

Esta função pode ser chamada imediatamente quer após criada a consulta parametrizável em DatabasePrepare() quer após redefinir a consulta para o status inicial com a ajuda do DatabaseReset().

Use esta função juntamente com o DatabaseReset(), para executar a consulta quantas vezes forem necessárias com diferentes valores de parâmetros.

A função é projetada para trabalhar com parâmetros de tipos simples. Se o parâmetro precisar mapear a matriz, use a função DatabaseBindArray().

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- lembramo-nos da hora de início antes de receber ticks
   uint start=GetTickCount();
//--- consultamos o histórico de ticks do dia
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
      PrintFormat("%s: CopyTicksRange(%s - %s) failed, error=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
      //--- quantidade de ticks obtidos em determinado intervalo de tempo
      PrintFormat("%s: CopyTicksRange received %d ticks in %d ms (from %s to %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- criamos um nome de arquivo para armazenar o banco de dados
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // o caractere ":" não é permitido nos nomes dos
//--- abrimos/criamos o banco de dados na pasta do terminal compartilhado
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("Database: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");

//--- criamos a tabela TICKS
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                       "SYMBOL             CHAR(10),"
                       "TIME               INT NOT NULL,"
                       "BID                REAL,"
                       "ASK                REAL,"
                       "LAST               REAL,"
                       "VOLUME             INT,"
                       "TIME_MSC           INT,"
```

```
                        "VOLUME_REAL        REAL);"))
     {
      Print("DB: ", filename, " create table TICKS failed with code ", GetLastError())
      DatabaseClose(db);
      return;
     }
//--- exibimos uma lista contendo todos os campos na tabela TICKS
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }
//--- criamos a consulta parametrizada para adicionar ticks à tabela TICKS
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // parâmetro de consulta
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }
//--- definimos o valor do primeiro parâmetro da consulta
   DatabaseBind(request, 0, _Symbol);
//--- lembramo-nos da hora de início antes de adicionar ticks à tabela TICKS
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- definimos o valor dos restantes parâmetros antes de adicionar um registro
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      //--- se a chamada anterior de DatabaseBind() foi bem-sucedida, definimos o segu
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      }
    if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

    //--- executamos a consulta para inserir o registro e verificamos se há erros
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
    //--- redefinimos a consulta para seu status inicial antes da seguinte atualizac
    if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
```

© 2000-2024, MetaQuotes Ltd.

```
          DatabaseFinalize(request);
          request_error=true;
          break;
         }
      } //--- acabamos por aqui, passamos por todos os ticks

//--- resultado da transação
   if(request_error)
     {
      PrintFormat("Table TICKS: failed to add %d ticks ", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table TICKS: added %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- fechamos o arquivo contendo o banco de dados e relatamos isso
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
/*
 Resultado:
  EURUSD: CopyTicksRange received 268061 ticks in 47 ms (from 2020.03.18 12:40 to 2020
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite opened successfully
  #| cid name        type     notnull dflt_value pk
  -+-----------------------------------------------
  1|   0 SYMBOL      CHAR(10)       0             0
  2|   1 TIME        INT            1             0
  3|   2 BID         REAL           0             0
  4|   3 ASK         REAL           0             0
  5|   4 LAST        REAL           0             0
  6|   5 VOLUME      INT            0             0
  7|   6 TIME_MSC    INT            0             0
  8|   7 VOLUME_REAL REAL           0             0
  Table TICKS: added 268061 ticks in 797 ms
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite created and closed
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
*/
```

Veja também

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

© 2000-2024, MetaQuotes Ltd.

DatabaseBindArray

Define a matriz como um valor de parâmetro.

```
bool  DatabaseBind(
   int  request,       // manipulador da consulta criada em DatabasePrepare
   int  index,         // índice do parâmetro na consulta
   T&   array[]        // valor de parâmetro como uma matriz
   );
```

Parâmetros

request

[in] Identificador da consulta criada em DatabasePrepare().

index

[in] Índice do parâmetro na consulta para o qual é necessário definir o valor. A numeração começa do zero.

array[]

[in] Matriz que deve ser definida como valor do parâmetro da consulta.

Valor retornado

Retorna true em caso de sucesso, caso contrário, false. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - tipo não suportado;
- ERR_ARRAY_BAD_SIZE (4011) - tamanho de matriz em bytes excede o valor INT_MAX;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_NOT_READY (5128) - não se pode usar a função para consulta no momento (a consulta está sendo realizada ou já foi concluída, é necessário chamar DatabaseReset).

Observação

Esta função deve ser usada caso a consulta SQL contenha os valores parametrizáveis "?" ou"?N", onde N é o número do parâmetro (começando de um). Além disso, a indexação de parâmetros no DatabaseBindArray() começa do zero.

Por exemplo:

```
INSERT INTO table VALUES (?,?,?)
```

Esta função pode ser chamada imediatamente quer após criada a consulta parametrizável em DatabasePrepare() quer após redefinir a consulta para o status inicial com a ajuda do DatabaseReset().

Use esta função juntamente com o DatabaseReset(), para executar a consulta quantas vezes forem necessárias com diferentes valores de parâmetros.

Exemplo:

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

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
//--- abrimos a caixa de diálogo para selecionar arquivos com extensão DAT
   string selected_files[];
   if(!FileSelectDialog("Selecione arquivos para carregamento", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Files not selected. Exit");
      return;
     }
//--- obtemos os parâmetros de arquivos
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i], filehandle[i
         total_size+=(double)filesize[i];
        }
     }
//--- verificamos o tamanho total do arquivo
   if(total_size==0)
     {
      PrintFormat("Total files size is 0. Exit");
      return;
     }

//--- criamos ou abrimos um banco de dados na pasta compartilhada do terminal
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- se a tabela FILES existir, devemos exclui-la
   if(DatabaseTableExists(db, "FILES"))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      //--- excluímos a tabela
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
        {
         Print("Failed to drop table FILES with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- criamos a tabela FILES
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                       "NAME           TEXT NOT NULL,"
                       "SIZE           INT  NOT NULL,"
                       "PERCENT_SIZE   REAL NOT NULL,"
                       "DATA           BLOB NOT NULL);"))
     {
      Print("DB: failed to create table FILES with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- exibimos uma lista contendo todos os campos na tabela FILES
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }

//--- criamos a consulta parametrizada para adicionar arquivos à tabela FILES
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // parâmetros de consulta
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- percorremos todos os arquivos e os adicionamos à tabela FILES
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
     {
      if(filehandle[i]!=INVALID_HANDLE)
        {
         char data[];
```

© 2000-2024, MetaQuotes Ltd.

```
        size=FileReadArray(filehandle[i], data);
        if(size==0)
          {
           PrintFormat("FileReadArray(%s) failed with code %d", selected_files[i], Ge
           continue;
          }

        count++;
        //--- definimos o valor dos parâmetros antes adicionar um arquivo à tabela
        if(!DatabaseBind(request, 0, selected_files[i]))
          {
           PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
           request_error=true;
           break;
          }
        if(!DatabaseBind(request, 1, size))
          {
           PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
           request_error=true;
           break;
          }
        if(!DatabaseBind(request, 2, double(size)*100./total_size))
          {
           PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
           request_error=true;
           break;
          }
        if(!DatabaseBindArray(request, 3, data))
          {
           PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
           request_error=true;
           break;
          }
        //--- executamos a consulta para inserir o registro e verificamos se há erros
        if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
          {
           PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
           DatabaseFinalize(request);
           request_error=true;
           break;
          }
        else
           PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
        //--- redefinimos a consulta para seu status inicial antes da seguinte atual
        if(!DatabaseReset(request))
          {
           PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
           DatabaseFinalize(request);
           request_error=true;
```

© 2000-2024, MetaQuotes Ltd.

```
            break;
           }
        }
     }
//--- resultado da transação
   if(request_error)
     {
      PrintFormat("Table FILES: failed to add %d files", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table FILES: added %d files", count);
     }

//--- fechamos o arquivo contendo o banco de dados e relatamos isso
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
```

Veja também

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

© 2000-2024, MetaQuotes Ltd.

DatabaseRead

Realiza o a transição para o próximo registro como resultado de uma consulta.

```
bool  DatabaseRead(
   int  request      // identificador de consulta recebido em DatabasePrepare
   );
```

Parâmetros

request

[in] Identificador de consulta recebido em DatabasePrepare().

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - nome da tabela não está especificado (string vazia ou NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta;
- ERR_DATABASE_NO_MORE_DATA (5126) - a tabela não existe (não é um erro, finalização normal).

Veja também

DatabasePrepare, DatabaseReadBind

© 2000-2024, MetaQuotes Ltd.

DatabaseReadBind

Realiza o a transição para o próximo registro e lê dados dele na estrutura.

```
bool  DatabaseReadBind(
   int    request,           // manipulador da consulta criada no DatabasePrepare
   void&  struct_object      // referência para a estrutura para leitura do registro
   );
```

Parâmetros

request

[in] Identificador da consulta criada em DatabasePrepare().

struct_object

[out] Referência para a estrutura em que serão lidos os dados do registro atual. A estrutura deve ter como membros apenas tipos numéricos e/ou strings (arrays não são permitidos) e não pode ser um herdeiro.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INVALID_PARAMETER (4003) - nome da tabela não está especificado (string vazia ou NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta;
- ERR_DATABASE_NO_MORE_DATA (5126) - a tabela não existe (não é um erro, finalização normal).

Observação

O número de campos na estrutura *struct_object* não pode exceder DatabaseColumnsCount(). Se o número de campos na estrutura *struct_object* for menor que o número de campos no registro, será realizada uma leitura parcial. Os dados restantes podem ser obtidos explicitamente usando as funções correspondentes DatabaseColumnText(), DatabaseColumnInteger() e assim por diante

Exemplo:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int db;
   string filename="company.sqlite";
//--- open
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATABASE_
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the table COMPANY exists then drop the table
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- delete the table
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("Failed to drop table COMPANY with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- create table
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- insert data
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
     {
      Print("DB: ", filename, " insert failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- prepare the request
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
```

© 2000-2024, MetaQuotes Ltd.

```
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- print records
   Person person;
   Print("Persons with salary > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ", person.addres
//--- delete request after use
   DatabaseFinalize(request);

   Print("Some statistics:");
//--- prepare new request about total salary
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("Total salary=", total_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);

//--- prepare new request about average salary
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " request failed with code ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
      Print("Average salary=", aver_salary);
     }
//--- delete request after use
   DatabaseFinalize(request);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- close database
   DatabaseClose(db);
  }
//+------------------------------------------------------------------
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

Veja também

DatabasePrepare, DatabaseRead

© 2000-2024, MetaQuotes Ltd.

DatabaseFinalize

Exclui uma consulta criada em DatabasePrepare().

```
void  DatabaseFinalize(
   int  request      // identificador de consulta recebido em DatabasePrepare
   );
```

Parâmetros

request

[in] Identificador de consulta recebido em DatabasePrepare().

Valor retornado

Não.

Observação

No caso de um identificador válido, a função mostrará o erro ERR_DATABASE_INVALID_HANDLE. O erro pode ser verificado usando GetLastError().

Veja também

DatabasePrepare, DatabaseExecute

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionBegin

Начинает выполнение транзакции.

```
bool  DatabaseTransactionBegin(
   int  database      // identificador de banco de dados recebido em DatabaseOpen
   );
```

Parâmetros

database

[in] Identificador de banco de dados recebido em DatabaseOpen().

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_INVALID_PARAMETER (4003) - parâmetro sql contendo string vazia;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta.

Observação

A função DatabaseTransactionBegin() deve ser chamada antes que realizar uma transação. Qualquer transação deve começar com uma chamada de DatabaseTransactionBegin() e terminar com uma chamada de DatabaseTransactionCommit().

Example:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- open/create the database in the common terminal folder
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
//--- if the DEALS table already exists, delete it
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- create the DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  request the entire trading history
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- request the history of deals in the specified interval
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("Deals in the trading history: %d ", deals_total);

//--- measure the transaction execution speed using DatabaseTransactionBegin/Database
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- delete the DEALS table, and then create it again
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- create a new DEALS table
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- test again, this time without using DatabaseTransactionBegin/DatabaseTransactio
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- report gain in time
   PrintFormat("Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acc
//--- close the database
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
```

© 2000-2024, MetaQuotes Ltd.

```
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration by
*/
//+------------------------------------------------------------------+
//| Deletes a table with the specified name from the database        |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop table DEALS with code ", GetLastError());
      return(false);
     }
//--- the table has been successfully deleted
   return(true);
  }
//+------------------------------------------------------------------+
//| Creates the DEALS table                                          |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- check if the table exists
   if(!DatabaseTableExists(database, "DEALS"))
      //--- create the table
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: create table failed with code ", GetLastError());
         return(false);
        }
//--- the table has been successfully created
   return(true);
  }
//+------------------------------------------------------------------+
//| Adds deals to the database table                                 |
//+------------------------------------------------------------------+
```

```
bool InsertDeals(int database, bool begintransaction=true)
  {
//--- Auxiliary variables
   ulong    deal_ticket;         // deal ticket
   long     order_ticket;        // the ticket of the order by which the deal was exec
   long     position_ticket;     // ID of the position to which the deal belongs
   datetime time;                // deal execution time
   long     type ;               // deal type
   long     entry ;              // deal direction
   string   symbol;              // the symbol fro which the deal was executed
   double   volume;              // operation volume
   double   price;               // price
   double   profit;              // financial result
   double   swap;                // swap
   double   commission;          // commission
   long     magic;               // Magic number
   long     reason;              // deal execution reason or source
//--- go through all deals and add to the database
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      // --- lock the database before executing transactions
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- add each deal using the following request
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tim
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__, deal_t
```

```
          PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
          failed=true;
          break;
         }
      }
//--- check for transaction execution errors
   if(failed)
     {
      //--- if fast transaction performance method is used
      if(begintransaction)
        {
         //--- roll back all transactions and unlock the database
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- if fast transaction performance method is used
   if(begintransaction)
     {
      //--- all transactions have been performed successfully - record changes and unl
      DatabaseTransactionCommit(database);
     }
//--- successful completion
   return(true);
  }
//+------------------------------------------------------------------+
```

Veja também

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionCommit

Conclui a execução da transação.

```
bool  DatabaseTransactionCommit(
   int  database      // identificador de banco de dados recebido em DatabaseOpen
   );
```

Parâmetros

database

[in] Identificador de banco de dados recebido em DatabaseOpen().

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_INVALID_PARAMETER (4003) - parâmetro sql contendo string vazia;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta.

Observação

A função DatabaseTransactionCommit() conclui todas as transações que foram executadas após a chamada da função DatabaseBeginTransaction(). Para uma conclusão bem-sucedida, toda transação deve começar com uma chamada de DatabaseTransactionBegin() e terminar com uma chamada de DatabaseTransactionCommit().

Veja também

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

© 2000-2024, MetaQuotes Ltd.

DatabaseTransactionRollback

Выполняет откат транзакций.

```
bool  DatabaseTransactionRollback(
   int  database      // identificador de banco de dados recebido em DatabaseOpen
   );
```

Parâmetros

database

[in] Identificador de banco de dados recebido em DatabaseOpen().

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_INTERNAL_ERROR (4001) - erro crítico de tempo de execução do sistema;
- ERR_INVALID_PARAMETER (4003) - parâmetro sql contendo string vazia;
- ERR_NOT_ENOUGH_MEMORY (4004) - memória insuficiente;
- ERR_WRONG_STRING_PARAMETER (5040) - erro ao converter a consulta em uma string UTF-8;
- ERR_DATABASE_INTERNAL (5120) - erro interno do banco de dados;
- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de banco de dados inválido;
- ERR_DATABASE_EXECUTE (5124) - erro de execução de consulta;

Observação

A chamada de DatabaseTransactionRollback() cancela todas as transações que foram executadas após a chamada da função DatabaseTransactionBegin (). A função DatabaseTransactionRollback() é necessária para reverter as alterações no banco de dados se ocorrerem erros durante a transação.

Veja também

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnsCount

Obtém o número de campos na consulta.

```
int  DatabaseColumnsCount(
   int  request      // identificador de consulta recebido em DatabasePrepare
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

Valor retornado

Número de campos ou -1 em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) - identificador de consulta inválido.

Observação

Para obter o número de campos de consulta criados em DatabasePrepare(), não é necessário chamar a função DatabaseRead(). Para outras funções do tipo DatabaseColumnXXXX(), primeiro você precisa chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseFinalize, DatabaseClose

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnName

Получает имя поля по номеру.

```
bool  DatabaseColumnName(
   int      request,     // identificador de consulta recebido em DatabasePrepare
   int      column,      // número do campo na consulta
   string&  name         // link para a variável para obter o nome do campo
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

nome

[out] Variável para registrar o nome do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) – índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request *pelo menos tiver sido feita uma chamada de* DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnType

Получает тип поля по номеру.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,      // identificador de consulta recebido em DatabasePrepare
   int  column        // número do campo na consulta
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

Valor retornado

Возвращает тип поля из перечисления ENUM_DATABASE_FIELD_TYPE. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) – índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request *pelo menos tiver sido feita uma chamada de* DatabaseRead().

ENUM_DATABASE_FIELD_TYPE

Identificador	Descrição
DATABASE_FIELD_TYPE_INVALID	Erro ao obter o tipo, o código do erro pode ser obtido usando int GetLastError()
DATABASE_FIELD_TYPE_INTEGER	Tipo inteiro
DATABASE_FIELD_TYPE_FLOAT	Tipo real
DATABASE_FIELD_TYPE_TEXT	Tipo string
DATABASE_FIELD_TYPE_BLOB	Tipo binário
DATABASE_FIELD_TYPE_NULL	Tipo especial NULL

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnSize

Получает размер поля в байтах.

```
int  DatabaseColumnSize(
   int  request,      // identificador de consulta recebido em DatabasePrepare
   int  column        // número do campo na consulta
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

Valor retornado

Se for bem-sucedido, retorna o tamanho do campo em bytes, caso contrário, -1. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) – índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request *pelo menos tiver sido feita uma chamada de* DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnText

Obtém o valor do campo do registro atual como uma string.

```
bool  DatabaseColumnText(
   int      request,     // identificador de consulta recebido em DatabasePrepare
   int      column,      // número do campo na consulta
   string&  value        // link para a variável para obter o valor
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

value

[out] Link para a variável para registrar o valor do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) - índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request pelo menos tiver sido feita uma chamada de DatabaseRead().

Para ler os valores do próximo registro, você deve primeiro chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnInteger

Obtém um valor int do registro atual.

```
bool  DatabaseColumnInteger(
   int   request,      // identificador de consulta recebido em DatabasePrepare
   int   column,       // número do campo na consulta
   int&  value         // link para a variável para obter o valor
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

value

[out] Link para a variável para registrar o valor do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) - índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request pelo menos tiver sido feita uma chamada de DatabaseRead().

Para ler os valores do próximo registro, você deve primeiro chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnLong

Obtém um valor long do registro atual.

```
bool  DatabaseColumnLong(
   int    request,     // identificador de consulta recebido em DatabasePrepare
   int    column,      // número do campo na consulta
   long&  value        // link para a variável para obter o valor
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

value

[out] Link para a variável para registrar o valor do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) - índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request pelo menos tiver sido feita uma chamada de DatabaseRead().

Para ler os valores do próximo registro, você deve primeiro chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnDouble

Obtém um valor double do registro atual.

```
bool  DatabaseColumnDouble(
   int      request,     // identificador de consulta recebido em DatabasePrepare
   int      column,      // número do campo na consulta
   double&  value        // link para a variável para obter o valor
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

value

[out] Link para a variável para registrar o valor do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) - índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request pelo menos tiver sido feita uma chamada de DatabaseRead().

Para ler os valores do próximo registro, você deve primeiro chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

DatabaseColumnBlob

Obtém o valor do campo do registro atual como um array.

```
bool  DatabaseColumnBlob(
   int    request,     // identificador de consulta recebido em DatabasePrepare
   int    column,      // número do campo na consulta
   void&  data[]       // link para a variável para obter o valor
   );
```

Parâmetros

request

[in] Identificador da consulta recebido em DatabasePrepare().

column

[in] Número do campo na consulta. A numeração dos campos começa em zero e não pode exceder o valor DatabaseColumnsCount() - 1.

data[]

[out] Link para um array para registrar o valor do campo.

Valor retornado

Retorna true em caso de sucesso ou false, em caso de erro. Para obter o código de erro, use GetLastError(), possíveis respostas:

- ERR_DATABASE_INVALID_HANDLE (5121) – identificador de consulta inválido;
- ERR_DATABASE_NO_MORE_DATA (5126) - índice column maior do que o valor DatabaseColumnsCount() -1.

Observação

O valor pode ser obtido apenas se antes para a consulta request pelo menos tiver sido feita uma chamada de DatabaseRead().

Para ler os valores do próximo registro, você deve primeiro chamar DatabaseRead().

Veja também

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2024, MetaQuotes Ltd.

Trabalho com DirectX

Funções e sombreadores do DirectX 11 são projetados para visualização em 3D diretamente no gráfico de preços.

Para criar gráficos tridimensionais, é necessário criar um contexto gráfico (DXContextCreate) com o tamanho de imagem desejado. Em seguida, prepare os buffers de vértice e de índice (DXBufferCreate) e crie os sombreadores de vértice e de pixel (DXShaderCreate). Para isso, basta exibir gráficos coloridos.

Para o próximo nível de gráficos, é necessário adicionar parâmetros de entrada (DXInputSet), destinados a transferir parâmetros adicionais de renderização para os sombreadores. Isso permite definir a posição da câmera e do objeto 3D, descrever as fontes de luz e implementar o controle usando o mouse e o teclado.

Desta maneira, as funções criadas diretamente em MQL5 permitem criar gráficos 3D animados diretamente no terminal MetaTrader 5 e não requerem ferramentas de terceiros para tal. Para que as funções funcionem, a placa de vídeo deve suportar DX 11 e sombreadores versão 5.0.

Para começar a trabalhar com a biblioteca, basta ler o artigo Como criar gráfico 3D em DirectX no MetaTrader 5.

Função	Ação
DXContextCreate	Cria um contexto gráfico para renderizar quadros do tamanho especificado
DXContextSetSize	Altera o tamanho do quadro do contexto gráfico criado em DXContextCreate()
DXContextSetSize	Obtém o tamanho do quadro do contexto gráfico criado em DXContextCreate().
DXContextClearColors	Define, para o buffer de renderização, todos os pixeis na cor especificada
DXContextClearDepth	Limpa o buffer de profundidade
DXContextGetColors	Obtém, do contexto gráfico, uma imagem com o tamanho e o deslocamento especificados
DXContextGetDepth	Obtém o buffer de profundidade do quadro renderizado
DXBufferCreate	Cria um buffer com tipo especificado baseado em uma matriz de dados
DXTextureCreate	Cria uma textura bidimensional, a partir de um retângulo com determinado tamanho cortado da imagem transferida
DXInputCreate	Cria os parâmetros de entrada do sombreador
DXInputSet	Define os parâmetros de entrada do sombreador
DXShaderCreate	Cria um sombreador com tipo especificado
DXShaderSetLayout	Define o layout do vértice para o sombreador de vértice

© 2000-2024, MetaQuotes Ltd.

Função	Ação
DXShaderInputsSet	Define os parâmetros de entrada do sombreador
DXShaderTexturesSet	Define a textura do sombreador.
DXDraw	Desenha os vértices do buffer de vértice definido em DXBufferSet()
DXDrawIndexed	Desenha as primitivas gráficas descritas pelo buffer de índice de DXBufferSet()
DXPrimiveTopologySet	Define o tipo de primitivas para renderização usando DXDrawIndexed()
DXBufferSet	Define o buffer para a renderização atual
DXShaderSet	Define o sombreador para renderização
DXHandleType	Retorna o tipo de identificador
DXRelease	Libera o identificador

© 2000-2024, MetaQuotes Ltd.

DXContextCreate

Cria um contexto gráfico para renderizar quadros com o tamanho especificado.

```
int  DXContextCreate(
   uint  width,       // largura em pixeis
   uint  height       // altura em pixeis
   );
```

Parâmetros

width

[in] Largura do quadro em pixeis.

height

[in] Altura do quadro em pixeis

Valor retornado

Identificador para o contexto criado ou, em caso de erro, INVALID_HANDLE. Para obter o código de erro, é necessário chamar a função GetLastError().

Observação

Todos os objetos gráficos criados com ajuda das funções DXBufferCreate, DXInputCreate, DXShaderCreate e DXTextureCreate só podem ser usados no contexto gráfico em que foram criados.

O tamanho do quadro pode ser alterado em DXContextSetSize().

Ó identificador criado que não é mais usado deve ser liberado explicitamente com ajuda da função DXRelease().

© 2000-2024, MetaQuotes Ltd.

DXContextSetSize

Altera o tamanho do quadro do contexto gráfico criado em DXContextCreate().

```
bool  DXContextSetSize(
   int    context,      // identifica o contexto gráfico
   uint&  width,        // largura em pixeis
   uint&  height        // altura em pixeis
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

width

[in] Largura do quadro em pixeis.

height

[in] Altura do quadro em pixeis.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O tamanho do quadro do contexto gráfico deve ser alterado apenas entre as renderizações do quadro.

© 2000-2024, MetaQuotes Ltd.

DXContextGetSize

Obtém o tamanho do quadro do contexto gráfico criado em DXContextCreate().

```
bool  DXContextGetSize(
   int    context,       // identifica o contexto gráfico
   uint&  width,         // largura em pixeis
   uint&  height         // altura em pixeis
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

width

[out] Largura do quadro em pixels.

height

[out] Altura do quadro em pixeis.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXContextClearColors

Define, para o buffer de renderização, todos os pixeis com a cor especificada.

```
bool  DXContextClearColors(
   int              context,      // identifica o contexto gráfico
   const DXVector&  color         // cor
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

color

[in] Cor para renderização.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

A função DXContextClearColors() pode ser usada para limpar o buffer de cores antes de renderizar o próximo quadro.

© 2000-2024, MetaQuotes Ltd.

DXContextClearDepth

Limpa o buffer de profundidade.

```
bool  DXContextClearDepth(
   int  context      // identifica o contexto gráfico
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

A função DXContextClearDepth() pode ser usada para limpar o buffer de profundidade antes de renderizar o próximo quadro.

© 2000-2024, MetaQuotes Ltd.

DXContextGetColors

Obtém, do contexto gráfico, uma imagem com o tamanho e o deslocamento especificados.

```
bool  DXContextGetColors(
   int    context,                      // identifica o contexto gráfico
   uint&  image[],                      // matriz de piexeis da imagem
   int    image_width=WHOLE_ARRAY,      // largura da imagem em pixeis
   int    image_height=WHOLE_ARRAY,     // altura da imagem em pixeis
   int    image_offset_x=0,             // deslocamento ao longo do eixo X
   int    image_offset_y=0              // deslocamento ao longo do eixo Y
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

image

[out] Matriz de pixeis com tamanho *image_width*image_height* em formato ARGB.

image_width=WHOLE_ARRAY

[in] Largura da imagem em pixeis.

image_height=WHOLE_ARRAY

[in] Altura da imagem em pixeis.

image_offset_x=0

[in] Deslocamento horizontal.

image_offset_y=0

[in] Deslocamento vertical.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXContextGetDepth

Obtém o buffer de profundidade do quadro renderizado.

```
bool  DXContextGetDepth(
   int     context,      // identifica o contexto gráfico
   float&  image[]       // matriz de valores de profundidade
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

image

[out] Matriz de valores do buffer de profundidade do quadro renderizado.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O buffer retornado contém em unidades relativas (de 0,0 a 1,0) a profundidade de cada pixel do quadro renderizado que pode ser obtido em DXContextGetColors().

© 2000-2024, MetaQuotes Ltd.

DXBufferCreate

Cria um buffer com tipo especificado baseado em uma matriz de dados.

```
int  DXBufferCreate(
   int                  context,              // identifica o contexto gráfico
   ENUM_DX_BUFFER_TYPE  buffer_type,          // tipo de buffer a ser criado
   const void&          data[],               // dados para o buffer
   uint                 start=0,              // índice inicial
   uint                 count=WHOLE_ARRAY     // número de elementos
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

buffer_type

[in] Tipo de buffer da enumeração ENUM_DX_BUFFER_TYPE.

data[]

[in] Dados para criar o buffer.

start

[in] Índice do primeiro elemento da matriz, a partir do qual são tomados os valores da matriz para criar o buffer. Por padrão, os dados são tomados desde o início da matriz.

count

[in] Número de valores. Por padrão, é usada toda a matriz (count=WHOLE_ARRAY).

Valor retornado

Identificador para o buffer criado ou, em caso de erro, INVALID_HANDLE. Para obter o código de erro, é necessário chamar a função GetLastError().

Observação

Para o buffer de índice, a matriz *data[]* deve ser do tipo uint, enquanto para o buffer de vértices é transferida uma matriz de estruturas que descreve vértices.

Ó identificador criado que não é mais usado deve ser liberado explicitamente com ajuda da função DXRelease().

ENUM_DX_BUFFER_TYPE

Identificador	Valor	Descrição
DX_BUFFER_VERTEX	1	Buffer de vértice
DX_BUFFER_INDEX	2	Buffer de índice

© 2000-2024, MetaQuotes Ltd.

DXTextureCreate

Cria uma textura bidimensional, a partir de um retângulo com determinado tamanho cortado da imagem transferida.

```
int  DXTextureCreate(
   int             context,         // identifica o contexto gráfico
   ENUM_DX_FORMAT  format,          // formato de cor do pixel
   uint            width,           // largura da imagem de origem
   uint            height,          // altura da imagem de origem
   const void&     data[],          // matriz de pixeis da imagem de origem
   uint            data_x,          // coordenada X do retângulo para criação da textu
   uint            data_y,          // coordenada Y do retângulo para criação da textu
   uint            data_width,      // largura do retângulo para criar a textura
   uint            data_height      // altura do retângulo para criar a textura
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

format

[in] Formato da cor dos pixeis especificado na enumeração ENUM_DX_FORMAT.

width

[in] Largura da imagem com base na qual é criada a textura.

height

[in] Altura da imagem com base na qual é criada a textura.

data

[in] Matriz de pixeis da imagem com base na qual é criada a textura.

data_x

[in] Coordenada X do retângulo (deslocamento horizontal) a partir do qual é criada a textura.

data_y

[in] Coordenada Y do retângulo (deslocamento vertical) a partir do qual é criada a textura.

data_width

[in] Largura do retângulo a partir do qual é criada a textura.

data_height

[in] Altura do retângulo a partir do qual é criada a textura.

Valor retornado

Identificador para a textura ou, em caso de erro, INVALID_HANDLE. Para obter o código de erro, é necessário chamar a função GetLastError().

Observação

© 2000-2024, MetaQuotes Ltd.

Ó identificador criado que não é mais usado deve ser liberado explicitamente com ajuda da função DXRelease().

ENUM_DX_FORMAT

Identificador	Valor	Em conformidade com DXGI_FORMAT
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Em conformidade com DXGI_FORMAT
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Em conformidade com DXGI_FORMAT
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Em conformidade com DXGI_FORMAT
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11

© 2000-2024, MetaQuotes Ltd.

Identificador	Valor	Em conformidade com DXGI_FORMAT
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

© 2000-2024, MetaQuotes Ltd.

DXInputCreate

Cria os parâmetros de entrada do sombreador.

```
int  DXInputCreate(
   int   context,        // identifica o contexto gráfico
   uint  input_size      // tamanho dos parâmetros de entrada em bytes
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

input_size

[in] Tamanho da estrutura do parâmetro em bytes.

Valor retornado

Identificador para os parâmetros de entrada do sombreador ou, em caso de erro, INVALID_HANDLE. Para obter o código de erro, é necessário chamar a função GetLastError().

Ó identificador criado que não é mais usado deve ser liberado explicitamente com ajuda da função DXRelease().

© 2000-2024, MetaQuotes Ltd.

DXInputSet

Define os parâmetros de entrada do sombreador.

```
bool  DXInputSet(
   int          input,       // identifica o contexto gráfico
   const void&  data         // dados para definição
   );
```

Parâmetros

input

[in] Identificador de parâmetros de entrada para o sombreador recebido em DXInputCreate().

data

[in] Dados para definir os parâmetros de entrada do sombreador.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

© 2000-2024, MetaQuotes Ltd.

DXShaderCreate

Cria um sombreador com tipo especificado.

```
int  DXShaderCreate(
   int                  context,            // identifica o contexto gráfico
   ENUM_DX_SHADER_TYPE  shader_type,        // tipo de sombreador
   const string         source,             // código fonte do sombreador
   const string         entry_point,        // ponto de entrada
   string&              compile_error       // cadeia de caracteres para receber mensa
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

shader_type

[out] Valor da enumeração ENUM_DX_SHADER_TYPE.

source

[in] Código fonte do sombreador descrito em HLSL 5.

entry_point

[in] Ponto de entrada, isto é, o nome da função no código fonte.

compile_error

[in] String para obter erros de compilação.

Valor retornado

Identificador para o sombreador ou, em caso de erro, INVALID_HANDLE. Para obter o código de erro, é necessário chamar a função GetLastError().

Observação

Ó identificador criado que não é mais usado deve ser liberado explicitamente com ajuda da função DXRelease().

ENUM_DX_SHADER_TYPE

Identificador	Valor	Descrição
DX_SHADER_VERTEX	0	Sombreador de vértice
DX_SHADER_GEOMETRY	1	Sombreador geométrico
DX_SHADER_PIXEL	2	Sombreador de pixel

© 2000-2024, MetaQuotes Ltd.

DXShaderSetLayout

Define o layout do vértice para o sombreador de vértice.

```
bool  DXShaderSetLayout(
   int                     shader,        // identificador do sombreador
   const DXVertexLayout&   layout[]       //
   );
```

Parâmetros

shader

[in] Identificador do sombreador de vértice criado em DXShaderCreate().

layout[]

[in] Matriz de descrição dos campos de vértices. A descrição é definida pela estrutura DXVertexLayout:

```
struct DXVertexLayout
  {
   string         semantic_name;       // The HLSL semantic associated with this e
   uint           semantic_index;      // The semantic index for the element. A se
   ENUM_DX_FORMAT format;              // The data type of the element data.
  };
```

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O layout deve corresponder ao tipo de vértices no buffer de vértice definido e ao tipo de entrada de vértices usados no ponto de entrada no código do sombreador de vértice.

O buffer de vértice para o sombreador é definido em DXBufferSet().

A estrutura DXVertexLayout é uma versão da estrutura D3D11_INPUT_ELEMENT_DESC do MSDN.

© 2000-2024, MetaQuotes Ltd.

DXShaderInputsSet

Define os parâmetros de entrada do sombreador.

```
bool  DXShaderInputsSet(
   int         shader,       // identificador do sombreador
   const int&  inputs[]      // matriz de identificadores de parâmetros de entrada
   );
```

Parâmetros

shader

[in] Identificador do sombreador criado em DXShaderCreate().

inputs[]

[in] Matriz de identificadores de parâmetros de entrada criados com ajuda de DXInputCreate().

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O tamanho da matriz de parâmetros de entrada deve ser igual ao número de objetos cbuffer declarados no código do sombreador.

© 2000-2024, MetaQuotes Ltd.

DXShaderTexturesSet

Define as texturas para o sombreador.

```
bool  DXShaderTexturesSet(
   int          shader,         // identificador do sombreador
   const  int&  textures[]      // matriz de identificadores de estruturas
   );
```

Parâmetros

shader

[in] Identificador do sombreador criado em DXShaderCreate().

textures[]

[in] Matriz de identificadores de textura criados com ajuda de DXTextureCreate().

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O tamanho da matriz de texturas deve ser igual ao número de objetos Texture2D declarados no código do sombreador.

© 2000-2024, MetaQuotes Ltd.

DXDraw

Desenha os vértices do buffer de vértice definido em DXBufferSet().

```
bool  DXDraw(
   int   context,               // identifica o contexto gráfico
   uint  start=0,               // índice do primeiro vértice
   uint  count=WHOLE_ARRAY      // número de vértices
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

start

[in] Índice do primeiro vértice para renderização.

count

[in] Número de vértices para renderização.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

Para renderização de vértices, os sombreadores devem primeiro ser definidos com ajuda de DXShaderSet().

© 2000-2024, MetaQuotes Ltd.

DXDrawIndexed

Desenha as primitivas gráficas descritas pelo buffer de índice de DXBufferSet().

```
bool  DXDrawIndexed(
   int   context,               // identifica o contexto gráfico
   uint  start=0,               // índice da primeira primitiva
   uint  count=WHOLE_ARRAY      // número de primitivas
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

start

[in] Índice da primeira primitiva para renderização.

count

[in] Número de primitivas para renderização.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

O tipo de primitivas descrito pelo buffer de índice é especificado com ajuda de DXPrimiveTopologySet().

Para renderização de primitivas, um buffer de vértice deve primeiro ser definido em DXBufferSet().

Para renderização de primitivas, os sombreadores devem primeiro ser definidos com ajuda de DXShaderSet().

© 2000-2024, MetaQuotes Ltd.

DXPrimiveTopologySet

Define o tipo de primitivas para renderização com ajuda de DXDrawIndexed().

```
bool  DXPrimiveTopologySet(
   int                          context,                // identifica o contexto gráfic
   ENUM_DX_PRIMITIVE_TOPOLOGY   primitive_topology      // tipo de primitiva
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

primitive_topology

[in] Valor da enumeração ENUM_DX_PRIMITIVE_TOPOLOGY.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

ENUM_DX_PRIMITIVE_TOPOLOGY

Identificador	Valor	Em conformidade com D3D11_PRIMITIVE_TOPOLOGY
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

© 2000-2024, MetaQuotes Ltd.

DXBufferSet

Define o buffer para a renderização atual.

```
bool  DXBufferSet(
   int   context,              // identifica o contexto gráfico
   int   buffer,               // identifica o buffer de vértice ou de índice
   uint  start=0,              // índice inicial
   uint  count=WHOLE_ARRAY     // número de elementos
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

buffer

[in] Identificador do buffer de vértice ou de índice criado em DXBufferCreate().

start

[in] Índice do primeiro elemento do buffer. Por padrão, são usados os dados do início do buffer.

count

[in] Número de valores a serem usados. Por padrão, todos os valores do buffer.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

A função DXBufferSet() deve ser chamada para definir os buffers de vértice e de índice para renderização com ajuda de DXDraw().

© 2000-2024, MetaQuotes Ltd.

DXShaderSet

Define o sombreador para renderização.

```
bool  DXShaderSet(
   int  context,       // identifica o contexto gráfico
   int  shader         // identificador do sombreador
   );
```

Parâmetros

context

[in] Identificador para o contexto gráfico criado em DXContextCreate().

shader

[in] Identificador do sombreador criado em DXShaderCreate().

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

Para renderização, podem ser simultaneamente usados vários sombreadores (de vértice, geométricos, de pixel).

© 2000-2024, MetaQuotes Ltd.

DXHandleType

Retorna o tipo de identificador.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // identificador
   );
```

Parâmetros

handle

[in] Identificador.

Valor retornado

Valor da enumeração ENUM_DX_HANDLE_TYPE

ENUM_DX_HANDLE_TYPE

Identificador	Valor	Descrição
DX_HANDLE_INVALID	0	Identificador inválido
DX_HANDLE_CONTEXT	1	Identificador para contexto gráfico
DX_HANDLE_SHADER	2	Identificador para sombreador
DX_HANDLE_BUFFER	3	Identificador para buffer de vértice ou de índice
DX_HANDLE_INPUT	4	Identificador para parâmetros de entrada do sombreador
DX_HANDLE_TEXTURE	5	Identificador para textura

© 2000-2024, MetaQuotes Ltd.

DXRelease

Libera o identificador.

```
bool  DXRelease(
   int  handle      // identificador
   );
```

Parâmetros

context

[in] Identificador a ser liberado.

Valor retornado

Se for bem-sucedido, retorna true, caso contrário, false. Para obter o código do erro é preciso chamar a função GetLastError().

Observação

Todos os identificadores criados que não são usados mais devem ser liberados explicitamente com ajuda da função DXRelease().

© 2000-2024, MetaQuotes Ltd.

Módulo MetaTrader para integração com Python

A MQL5 foi projetada para desenvolver aplicativos de negociação de alto desempenho nos mercados financeiros e não possui análogos entre outras linguagens especializadas para algotrading. A sintaxe e a velocidade dos programas em MQL5 se aproximam mais das do C++, possui suporte a Opencl e integração com o MS Visual Studio, dispõe de bibliotecas de estatística, lógica difusa e ALGLIB. No ambiente de desenvolvimento do MetaEditor existe também um suporte nativo a bibliotecas .NET com a importação inteligente de funções sem escrever wrappers especiais, podem-se usar DLL de C++ de terceiros. Arquivos de códigos fonte em C++ (CPP e H) podem ser editados e compilados em DLLs diretamente do editor, basta usar o Microsoft Visual Studio instalado no computador do usuário.

Python é uma linguagem de programação moderna de alto nível para o desenvolvimento de scripts e aplicativos. Possui muitas bibliotecas para aprendizado de máquina, automação de processos, análise de dados e visualização.

O pacote MetaTrader para Python é projetado para o recebimento conveniente e rápido de informações de negociação através de comunicação entre processos diretamente do terminal MetaTrader 5. Os dados recebidos desta maneira podem ser utilizados para cálculos estatísticos e aprendizado de máquina.

Instalando o pacote na linha de comando:

```
pip install MetaTrader5
```

Atualizando o pacote na linha de comando:

```
pip install --upgrade MetaTrader5
```

Funções para integração de MetaTrader 5 e Python

Função	Ação
initialize	Estabelece a conexão com o terminal MetaTrader 5
login	Conecta-se a uma conta de negociação com os parâmetros especificados
shutdown	Fecha a conexão estabelecida anteriormente com o terminal MetaTrader 5
version	Retorna a versão do terminal MetaTrader 5
last_error	Retorna informações sobre o último erro
account_info	Obtém informações atuais sobre a conta de negociação
terminal_Info	Obtém o estado e os parâmetros do terminal MetaTrader 5 conectado
symbols_total	Obtém o número total de instrumentos financeiros no terminal MetaTrader 5
symbols_get	Obtém todos os instrumentos financeiros do terminal MetaTrader 5
symbol_info	Obtém informações sobre o instrumento financeiro especificado
symbol_info_tick	Obtém o último tick do instrumento financeiro especificado

© 2000-2024, MetaQuotes Ltd.

Função	Ação
symbol_select	Seleciona o símbolo na janela MarketWatch ou remove o símbolo deste janela
market_book_add	Faz com que o terminal MetaTrader 5 receba eventos sobre mudanças no livro de ofertas para o símbolo especificado
market_book_get	Retorna uma tupla desde BookInfo contendo os registros do livro de ofertas para o símbolo especificado
market_book_release	Cancela a subscrição do terminal MetaTrader 5 para receber eventos sobre alterações no livro de ofertas para o símbolo especificado
copy_rates_from	Recebe barras do terminal MetaTrader 5, a partir da data especificada
copy_rates_from_pos	Recebe barras do terminal MetaTrader 5, a partir do índice especificado
copyrates_range	Recebe barras a partir do terminal MetaTrader 5, no intervalo de datas especificado
copy_ticks_from	Recebe ticks do terminal MetaTrader 5, a partir da data especificada
copy_ticks_range	Recebe ticks a partir do terminal MetaTrader 5, no intervalo de datas especificado
orders_total	Obtém o número de ordens ativas.
orders_get	Obtém ordens ativas com a capacidade de filtrar por símbolo ou ticket
order_calc_margin	Retorna o tamanho da margem na moeda da conta para a operação de negociação especificada
order_calc_profit	Retorna o valor do lucro na moeda da conta para a operação de negociação especificada
order_check	Verifica que há fundos suficientes para realizar a operação de negociação requerida
order_send	Envia do terminal para o servidor de negociação uma solicitação para concluir uma operação de negociação.
positions_total	Obtém o número de posições abertas
positions_get	Obtém posições abertas com a capacidade de filtrar por símbolo ou bilhete
history_orders_total	Obtém o número de ordens no histórico de negociação no intervalo especificado
history_orders_get	Obtém ordens do histórico de negociação com a capacidade de filtrar por bilhete ou posição
history_deals_total	Obtém o número de transações no histórico de negociação no intervalo especificado

© 2000-2024, MetaQuotes Ltd.

Função	Ação
history_deals_get	Obtém transações do histórico de negociação com a capacidade de filtrar por bilhete ou posição

Exemplo de conexão do Python ao MetaTrader 5

1. Baixe a última versão do Python 3.8 da página https://www.python.org/downloads/windows
2. Ao instalar o Python, marque a caixa "Add Python 3.8 to PATH%" para poder executar scripts Python a partir da linha de comando.
3. Instale o módulo MetaTrader5 a partir da linha de comando

```
pip install MetaTrader5
```

4. Adicione os pacotes matplotlib e pandas

```
pip install matplotlib
pip install pandas
```

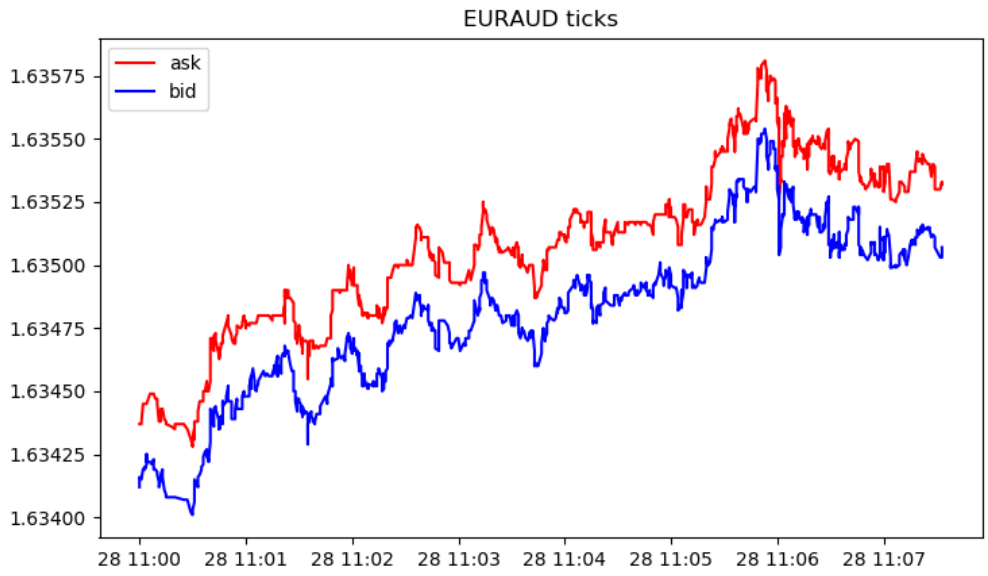
5. Execute o script de teste

```
from datetime import datetime
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# conecte-se ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# consultamos o estado e os parâmetros de conexão
print(mt5.terminal_info())
# obtemos informações sobre a versão do MetaTrader 5
print(mt5.version())

# solicitamos 1 000 ticks de EURAUD
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_T
# solicitamos ticks de AUDUSD no intervalo 2019.04.01 13:00 - 2019.04.02 13:00
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# obtemos barras de vários instrumentos de diferentes maneiras
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1, datetime(2020,1,28,13),
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1, datetime(2020,1,27,13)

# concluímos a conexão ao MetaTrader 5
mt5.shutdown()

#DATA
```

© 2000-2024, MetaQuotes Ltd.

```
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
for val in eurcad_rates[:10]: print(val)

#PLOT
# a partir dos dados recebidos criamos o DataFrame
ticks_frame = pd.DataFrame(euraud_ticks)
# plotamos os ticks no gráfico
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# exibimos rótulos
plt.legend(loc='upper left')

# adicionamos cabeçalho
plt.title('EURAUD ticks')

# mostramos o gráfico
plt.show()
```

6. Obtemos os dados e o gráfico



© 2000-2024, MetaQuotes Ltd.

```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
```

© 2000-2024, MetaQuotes Ltd.

```
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

© 2000-2024, MetaQuotes Ltd.

initialize

Estabelece a conexão com o terminal MetaTrader 5. Existem 3 variantes de chamada.

Chamada sem parâmetros. O terminal para conexão será encontrado automaticamente.

```
initialize()
```

Chamada indicando o caminho para o terminal MetaTrader 5 ao qual é preciso se conectar.

```
initialize(
   path                      // caminho para o arquivo EXE do terminal MetaTrader
   )
```

Uma chamada indicando o caminho e os parâmetros da conta de negociação.

```
initialize(
   path,                     // caminho para o arquivo EXE do terminal MetaTrader 5
   login=LOGIN,              // número da conta
   password="PASSWORD",      // senha
   server="SERVER",          // nome do servidor conforme definido no terminal
   timeout=TIMEOUT,          // tempo de espera esgotado
   portable=False            // modo portable
   )
```

Parâmetros

path

[in] Caminho para o arquivo metatrader.exe ou metatrader64.exe. Parâmetro opcional sem nome. É indicado primeiro sem nome de parâmetro. Se o caminho não for especificado, o módulo tentará encontrar arquivo executável por conta própria.

login=LOGIN

[in] Número da conta de negociação. Parâmetro nomeado opcional. Se não for especificado, será usada a última conta de negociação.

password="PASSWORD"

[in] Senha da conta de negociação. Parâmetro nomeado opcional. Se a senha não for especificada, será automaticamente colocada a senha armazenada no banco de dados do terminal para a conta de negociação especificada.

server="SERVER"

[in] Nome do servidor de negociação. Parâmetro nomeado opcional. Se a servidor não for especificado, será automaticamente colocada servidor armazenado no banco de dados do terminal para a conta de negociação especificada.

timeout=TIMEOUT

[in] Tempo limite de conexão em milissegundos. Parâmetro nomeado opcional. Se não especificado, será usado um valor de 60 000 (60 segundos).

portable=False

[in] Sinal de inicialização do terminal no modo portable. Parâmetro nomeado opcional. Se não especificado, será usado False.

© 2000-2024, MetaQuotes Ltd.

Valor retornado

Retorna True se a conexão com o terminal MetaTrader 5 for bem-sucedida, caso contrário, False.

Observação

Se necessário, se for chamado initialize(), o terminal MetaTrader 5 será inicializado para executar a conexão.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dads sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão com o terminal MetaTrader 5 para a conta especificada
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# imprimimos informações sobre o estado da conexão, o nome do servidor e a conta de ne
print(mt5.terminal_info())
# imprimimos informações sobre a versão do MetaTrader 5
print(mt5.version())

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

shutdown, terminal_info, version

© 2000-2024, MetaQuotes Ltd.

login

Conecta-se a uma conta de negociação com os parâmetros especificados.

```
login(
   login,                    // número da conta
   password="PASSWORD",      // senha
   server="SERVER",          // nome do servidor como definido no terminal
   timeout=TIMEOUT           // tempo de espera esgotado
   )
```

Parâmetros

login

[in] Número da conta de negociação. Parâmetro sem nome obrigatório.

password

[in] Senha da conta de negociação. Parâmetro nomeado opcional. Se a senha não for especificada, será automaticamente colocada a senha armazenada no banco de dados do terminal.

server

[in] Nome do servidor de negociação. Parâmetro nomeado opcional. Se o servidor não for especificado, será automaticamente colocado o último servidor.

timeout=TIMEOUT

[in] Tempo limite de conexão em milissegundos. Parâmetro nomeado opcional. Se não especificado, será usado um valor de 60 000 (60 segundos). Se a conexão não for estabelecida durante esse período, a chamada será forçosamente encerrada e será gerada sua remoção.

Valor retornado

True em caso de conexão bem-sucedida à conta de negociação, caso contrário, False.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# imprimimos informações sobre a versão do MetaTrader 5
print(mt5.version())
# conectamo-nos à conta de negociação sem especificar senha e servidor
account=17221085
authorized=mt5.login(account)  # a senha será retirada do banco de dados do terminal,
if authorized:
    print("connected to account #{}".format(account))
else:
```

© 2000-2024, MetaQuotes Ltd.

```
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# agora conectamo-nos a outra conta de negociação indicando senha
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # exibimos os dados sobre a conta de negociação como estão
    print(mt5.account_info())
    # exibimos os dados da conta de negociação como uma lista
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']

connected to account #17221085

connected to account #25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
account properties:
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
   margin_level=183084.6054809638
   margin_so_call=50.0
   margin_so_so=30.0
```

© 2000-2024, MetaQuotes Ltd.

```
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

Veja também

initialize, shutdown

© 2000-2024, MetaQuotes Ltd.

shutdown

Fecha a conexão estabelecida anteriormente com o terminal MetaTrader 5.

```
shutdown()
```

Valor retornado

Não.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed")
    quit()

# imprimimos informações sobre o estado da conexão, o nome do servidor e a conta de ne
print(mt5.terminal_info())
# imprimimos informações sobre a versão do MetaTrader 5
print(mt5.version())

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

initialize, login_py, terminal_info, version

© 2000-2024, MetaQuotes Ltd.

version

Retorna a versão do terminal MetaTrader 5.

```
version()
```

Valor retornado

Retorna o número da versão, o número da compilação e a data de lançamento do terminal MetaTrader 5. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função version() retorna o número da versão, o número da compilação e a data de lançamento do terminal como uma tupla de três valores:

Tipo	Descrição	Exemplo de valor
integer	Versão do terminal MetaTrader 5	500
integer	Compilação	2007
string	Data de lançamento da compilação	'25 Feb 2019'

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# imprimimos informações sobre a versão do MetaTrader 5
print(mt5.version())

# imprimimos informações sobre o estado da conexão, o nome do servidor e a conta de ne
print(mt5.terminal_info())
print()

# obtemos as propriedades na forma de um dicionário
terminal_info_dict=mt5.terminal_info()._asdict()
# convertemos o dicionário num DataFrame e imprimimos
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
```

© 2000-2024, MetaQuotes Ltd.

```
print(df[:-1])

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True, dlls_a

terminal_info() as dataframe:
                 property                        value
0       community_account                         True
1    community_connection                         True
2               connected                         True
3            dlls_allowed                        False
4           trade_allowed                        False
5       tradeapi_disabled                        False
6           email_enabled                        False
7             ftp_enabled                        False
8   notifications_enabled                        False
9                    mqid                        False
10                  build                         2367
11                maxbars                         5000
12               codepage                         1251
13              ping_last                        77881
14      community_balance                      707.107
15         retransmission                            0
16                company     MetaQuotes Software Corp.
17                   name                  MetaTrader 5
18               language                      Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

Veja também

initialize, shutdown, terminal_info

© 2000-2024, MetaQuotes Ltd.

last_error

Retorna informações sobre o último erro.

```
last_error()
```

Valor retornado

Retorna o código do último erro e sua descrição na forma de uma tupla.

Observação

last_error() permite que obter o código de erro em caso de execução mal sucedida de qualquer função da biblioteca MetaTrader 5. É semelhante a GetLastError(), mas são usados seus próprios códigos de erro. Valores possíveis:

Constante	Valor	Descrição
RES_S_OK	1	generic success
RES_E_FAIL	-1	generic fail
RES_E_INVALID_PARAMS	-2	invalid arguments/parameters
RES_E_NO_MEMORY	-3	no memory condition
RES_E_NOT_FOUND	-4	no history
RES_E_INVALID_VERSION	-5	invalid version
RES_E_AUTH_FAILED	-6	authorization failed
RES_E_UNSUPPORTED	-7	unsupported method
RES_E_AUTO_TRADING_DISABLED	-8	auto-trading disabled
RES_E_INTERNAL_FAIL	-10000	internal IPC general error
RES_E_INTERNAL_FAIL_SEND	-10001	internal IPC send failed
RES_E_INTERNAL_FAIL_RECEIVE	-10002	internal IPC recv failed
RES_E_INTERNAL_FAIL_INIT	-10003	internal IPC initialization fail
RES_E_INTERNAL_FAIL_CONNECT	-10003	internal IPC no ipc
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	internal timeout

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

version, GetLastError

© 2000-2024, MetaQuotes Ltd.

account_info

Obtém informações atuais sobre a conta de negociação atual.

```
account_info()
```

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função pode de uma chamada retornar todas as informações que podem ser obtidas usando AccountInfoInteger, AccountInfoDouble e AccountInfoString.

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# conectamo-nos à conta de negociação sem especificar senha e servidor
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # exibimos os dados sobre a conta de negociação como estão
        print(account_info)
        # exibimos os dados da conta de negociação como um dicionário
        print("Show account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

       # convertemos o dicionário num DataFrame e imprimimos
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("failed to connect to trade account 25115284 with password=gqz0343lbdm, erro

# concluímos a conexão ao terminal MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
Show account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
              property                     value
0                login                  25115284
1           trade_mode                         0
2             leverage                       100
3         limit_orders                       200
4       margin_so_mode                         0
5        trade_allowed                      True
6         trade_expert                      True
7          margin_mode                         2
8      currency_digits                         2
9           fifo_close                     False
10             balance                   99588.3
```

```
11              credit                          0
12              profit                     -45.13
13              equity                    99543.2
14              margin                      54.37
15         margin_free                    99488.8
16        margin_level                     183085
17      margin_so_call                         50
18        margin_so_so                         30
19      margin_initial                          0
20  margin_maintenance                          0
21              assets                          0
22         liabilities                          0
23  commission_blocked                          0
24                name                James Smith
25              server            MetaQuotes-Demo
26            currency                        USD
27             company  MetaQuotes Software Corp.
```

Veja também

initialize, shutdown, login

© 2000-2024, MetaQuotes Ltd.

terminal_info

Obtém o status e as configurações do terminal do cliente MetaTrader 5 conectado.

```
terminal_info()
```

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função retorna numa chamada todas as informações que podem ser obtidas usando TerminalInfoInteger, TerminalInfoDouble e TerminalInfoDouble.

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# imprimimos informações sobre a versão do MetaTrader 5
print(mt5.version())
# exibimos informações sobre as configurações e o status do terminal
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # exibimos os dados sobre o terminal tal qual como estão
    print(terminal_info)
    # exibimos os dados como uma lista
    print("Show terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
   # convertemos o dicionário num DataFrame e imprimimos
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
```

© 2000-2024, MetaQuotes Ltd.

```
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Show terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                 property                      value
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13              ping_last                      80953
14      community_balance                    707.107
```

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

Veja também

initialize, shutdown, version

© 2000-2024, MetaQuotes Ltd.

symbols_total

Obtém o número total de instrumentos financeiros no terminal MetaTrader 5.

```
symbols_total()
```

Valor retornado

Valor inteiro.

Observação

A função é semelhante a SymbolsTotal(), mas, além disso, retorna o número total de símbolos, incluindo os personalizados e os desabilitados no MarketWatch.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos o número de instrumentos financeiros
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("Symbols not found")

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

symbols_get, symbol_select, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbols_get

Obtém todos os instrumentos financeiros do terminal MetaTrader 5.

```
symbols_get(
   group="GROUP"      // filtro para selecionar símbolos
)
```

group="GROUP"

[in] Filtro para selecionar apenas o grupo de símbolos necessários. Parâmetro opcional. Se um grupo for especificado, a função retornará apenas os símbolos que satisfazem os critérios especificados.

Valor retornado

Retorna símbolos como uma tupla (tuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

O parâmetro *group* permite filtrar símbolos por nome, é permitido usar '*' no início e no final da linha.

O parâmetro *group* pode ser usado como parâmetro nomeado e como não nomeado. Ambas as variantes funcionam da mesma maneira. O uso de uma variante nomeada (group="GROUP") torna o código mais compreensível.

O parâmetro *group* pode conter várias condições, separadas por vírgulas. A condição pode ser especificada como uma máscara usando '*'. Para exclusões, pode-se usar o símbolo de negação lógica '!'. Neste caso, todas as condições são aplicadas sequencialmente, ou seja, primeiro deve-se especificar as condições para inclusão no grupo e, em seguida, a condição de exclusão. Por exemplo, *group*="*, !EUR" significa que primeiro é necessário selecionar todos os símbolos e, em seguida, excluir os que contêm "EUR" no nome.

Ao contrário de symbol_info(), a função symbols_get() retorna informações sobre todos os símbolos solicitados numa chamada.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos todos os símbolos
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
```

© 2000-2024, MetaQuotes Ltd.

```
# exibimos os 5 primeiros
for s in symbols:
    count+=1
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# obtemos símbolos cujos nomes contêm RU
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# obtemos símbolos cujos nomes não contêm USD, EUR, JPY e GBP
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Symbols:  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_dea
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
```

© 2000-2024, MetaQuotes Ltd.

```
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
```

Veja também

symbols_total, symbol_select, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbol_info

Obtém informações sobre o instrumento financeiro especificado.

```
symbol_info(
   symbol       // nome do instrumento financeiro
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função consegue de uma chamada retornar todas as informações que podem ser obtidas usando SymbolInfoInteger, SymbolInfoDouble e SymbolInfoString.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# tentamos ativar a exibição do símbolo EURJPY no MarketWatch
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Failed to select EURJPY")
    mt5.shutdown()
    quit()

# imprimimos as propriedades do símbolo EURJPY
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # exibimos os dados sobre o terminal tal qual como estão
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # exibimos as propriedades do símbolo como uma lista
    print("Show symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
```

© 2000-2024, MetaQuotes Ltd.

```
# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURJPY: spread = 17   digits = 3
Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
  last=0.0
```

© 2000-2024, MetaQuotes Ltd.

```
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
currency_profit=JPY
```

© 2000-2024, MetaQuotes Ltd.

```
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

Veja também

account_info, terminal_info

© 2000-2024, MetaQuotes Ltd.

symbol_info_tick

Obtém o último tick do instrumento financeiro especificado.

```
symbol_info_tick(
   symbol       // nome do instrumento financeiro
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

Valor retornado

Retorna informações na forma de uma tupla. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função é semelhante a SymbolInfoTick.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# tentamos ativar a exibição do símbolo GBPUSD no MarketWatch
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Failed to select GBPUSD")
    mt5.shutdown()
    quit()

# imprimimos o último tick do símbolo GBPUSD
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# imprimimos os valores dos campos de tick como uma lista
print("Show symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0, time_msc=158507033
Show symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

Veja também

symbol_info, symbol_info

© 2000-2024, MetaQuotes Ltd.

symbol_select

Seleciona o símbolo na janela MarketWatch ou remove o símbolo deste janela.

```
symbol_select(
   symbol,       // nome do instrumento financeiro
   enable=None   // ativar/desativar
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

enable

[in] Alternador. Parâmetro sem nome opcional. Se o valor for false, o símbolo deverá ser excluído da janela do MarketWatch, caso contrário, o símbolo deverá ser selecionado na janela do MarketWatch. O símbolo não poderá ser removido se houver gráficos abertos com este símbolo ou se houver posições abertas com base neste símbolo.

Valor retornado

true, em caso de execução bem-sucedida, caso contrário, false.

Observação

A função é semelhante a SymbolSelect.

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# tentamos ativar a exibição do símbolo EURCAD no MarketWatch
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("Failed to select EURCAD, error code =",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit =",sy
    print()

    # obtemos as propriedades do símbolo como um dicionário
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
```

© 2000-2024, MetaQuotes Ltd.

```
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

   # convertemos o dicionário num DataFrame e imprimimos
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
```

© 2000-2024, MetaQuotes Ltd.

```
option_mode=0
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
```

© 2000-2024, MetaQuotes Ltd.

```
  price_greeks_gamma=0.0
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() as dataframe:
         property                                  value
0          custom                                  False
1      chart_mode                                      0
2          select                                   True
3         visible                                   True
4   session_deals                                      0
..            ...                                    ...
91        formula
92           isin
93           name                                 EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                            Forex\EURCAD

[96 rows x 2 columns]
```

Veja também

symbol_info

© 2000-2024, MetaQuotes Ltd.

market_book_add

Faz com que o terminal MetaTrader 5 receba eventos sobre mudanças no livro de ofertas para o símbolo especificado.

```
market_book_add(
   symbol      // nome do instrumento financeiro
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

Valor retornado

true, em caso de execução bem-sucedida, caso contrário, false.

Observação

A função é análoga de MarketBookAdd.

Veja também

market_book_get, market_book_release, Estrutura do livro de ofertas

© 2000-2024, MetaQuotes Ltd.

market_book_get

Retorna uma tupla desde BookInfo contendo os registros do livro de ofertas para o símbolo especificado.

```
market_book_get(
   symbol      // nome do instrumento financeiro
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

Valor retornado

Retorna o conteúdo do livro de ofertas como uma tupla desde o log de BookInfo contendo o tipo de pedido, preço e volume em lotes. BookInfo é análogo à estrutura MqlBookInfo.

Retorna None em caso de erro, as informações sobre o erro podem ser obtidas usando last_error().

Observação

Anteriormente com ajuda da função market_book_add() era realizada a subscrição para receber eventos de mudanças no livro de ofertas.

A função é análoga a MarketBookGet.

Exemplo:

```
import MetaTrader5 as mt5
import time
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
   # concluímos a conexão ao terminal MetaTrader 5
    mt5.shutdown()
    quit()

# subscreva para receber atualizações no livro de ofertas para o símbolo EURUSD (Depth
if mt5.market_book_add('EURUSD'):
  # obtemos 10 vezes em um loop os dados do livro de ofertas
   for i in range(10):
        # obtemos o conteúdo do livro de ofertas (Depth of Market)
        items = mt5.market_book_get('EURUSD')
        # exibimos todo o livro de ofertas como uma string tal qual como está
        print(items)
        # agora exibimos cada solicitação separadamente para maior clareza
```

© 2000-2024, MetaQuotes Ltd.

```
        if items:
            for it in items:
                # conteúdo da solicitação
                print(it._asdict())
        # vamos fazer uma pausa de 5 segundos antes da próxima solicitação de dados do
        time.sleep(5)
  # cancelamos a subscrição de atualizações no livro de ofertas (Depth of Market)
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
```

© 2000-2024, MetaQuotes Ltd.

```
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

Veja também

market_book_add, market_book_release, Estrutura do livro de ofertas

© 2000-2024, MetaQuotes Ltd.

market_book_release

Cancela a subscrição do terminal MetaTrader 5 para receber eventos sobre alterações no livro de ofertas para o símbolo especificado.

```
market_book_release(
   symbol      // nome do instrumento financeiro
)
```

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

Valor retornado

true, em caso de execução bem-sucedida, caso contrário, false.

Observação

A função é análoga a MarketBookRelease.

Veja também

market_book_add, market_book_get, Estrutura do livro de ofertas

© 2000-2024, MetaQuotes Ltd.

copy_rates_from

Recebe barras do terminal MetaTrader 5, a partir da data especificada.

```
copy_rates_from(
   symbol,        // nome do símbolo
   timeframe,     // período gráfico
   date_from,     // data de abertura da barra inicial
   count          // número de barras
   )
```

Parâmetros

symbol

[in] Nome do instrumento financeiro, por exemplo, "EURUSD". Parâmetro sem nome obrigatório.

timeframe

[in] Período gráfico para o qual são solicitadas as barras. É definido pelo valor a partir da enumeração TIMEFRAME. Parâmetro sem nome obrigatório.

date_from

[in] Data de abertura da primeira barra a partir da amostra solicitada. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

count

[in] O número de barras que é necessário obter. Parâmetro sem nome obrigatório.

Valor retornado

Retorna barras na forma de arrays numpy contendo colunas nomeadas time, open, high, low, close, tick_volume, spread e real_volume. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

Para mais informações, veja a função CopyRates().

Somente dados cuja data seja menor (anterior) ou igual a data especificada são retornados. Isso significa que a hora de abertura de qualquer barra é sempre menor ou igual ao valor especificado.

O terminal MetaTrader 5 fornece barras apenas nos limites do histórico disponível para o usuário nos gráficos. O número de barras disponíveis para o usuário é definido nas configurações com o parâmetro "Máx. de barras no gráfico"

Python, ao criar um objeto datetime, usa o fuso horário local, enquanto o terminal MetaTrader 5 armazena o tempo de ticks e de abertura de barras no fuso horário UTC (sem deslocamento). Assim, para executar funções que usam tempo, deve-se criar um datetime no horário UTC. Os dados recebidos do terminal MetaTrader 5 têm hora UTC.

TIMEFRAME é uma listagem com os valores possíveis dos períodos do gráfico

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
TIMEFRAME_M1	1 minuto
TIMEFRAME_M2	2 minutos
TIMEFRAME_M3	3 minutos
TIMEFRAME_M4	4 minutos
TIMEFRAME_M5	5 minutos
TIMEFRAME_M6	6 minutos
TIMEFRAME_M10	10 minutos
TIMEFRAME_M12	12 minutos
TIMEFRAME_M12	15 minutos
TIMEFRAME_M20	20 minutos
TIMEFRAME_M30	30 minutos
TIMEFRAME_H1	1 hora
TIMEFRAME_H2	2 horas
TIMEFRAME_H3	3 horas
TIMEFRAME_H4	4 horas
TIMEFRAME_H6	6 horas
TIMEFRAME_H8	8 horas
TIMEFRAME_H12	12 horas
TIMEFRAME_D1	1 dia
TIMEFRAME_W1	1 semana
TIMEFRAME_MN1	1 mês

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos o módulo pandas para exibir os dados recebidos na forma de uma tabela
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas
pd.set_option('display.width', 1500)      # largura máxima da tabela
# importamos o módulo pytz para trabalhar com o fuso horário
```

© 2000-2024, MetaQuotes Ltd.

```
import pytz

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# definimos o fuso horário como UTC
timezone = pytz.timezone("Etc/UTC")
# criamos o objeto datetime no fuso horário UTC para que não seja aplicado o deslocame
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# recebemos 10 barras de EURUSD H4 a partir de 01/10/2019 no fuso horário UTC
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
# exibimos cada elemento de dados recebidos numa nova linha
print("Exibimos os dados recebidos como estão")
for rate in rates:
    print(rate)

# a partir dos dados recebidos criamos o DataFrame
rates_frame = pd.DataFrame(rates)
# convertemos o tempo em segundos no formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# exibimos dados
print("\nExibimos o dataframe com dados")
print(rates_frame)

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Exibimos os dados como eles são
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

Exibimos o dataframe com dados
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
```

```
1 2020-01-08 16:00:00  1.11199  1.11308  1.11086  1.11179        10641       1
2 2020-01-08 20:00:00  1.11178  1.11178  1.11016  1.11053         4806       1
3 2020-01-09 00:00:00  1.11053  1.11193  1.11033  1.11173         3480       1
4 2020-01-09 04:00:00  1.11173  1.11189  1.11126  1.11182         2236       1
5 2020-01-09 08:00:00  1.11181  1.11203  1.10983  1.10993         7984       1
6 2020-01-09 12:00:00  1.10994  1.11173  1.10965  1.11148         7406       1
7 2020-01-09 16:00:00  1.11149  1.11149  1.10923  1.11046         7468       1
8 2020-01-09 20:00:00  1.11046  1.11097  1.11033  1.11051         3450       1
9 2020-01-10 00:00:00  1.11051  1.11093  1.11017  1.11041         2448       1
```

Veja também

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_rates_from_pos

Recebe barras do terminal MetaTrader 5, a partir do índice especificado.

```
copy_rates_from_pos(
   symbol,        // nome do símbolo
   timeframe,     // período gráfico
   start_pos,     // número da barra inicial
   count          // número de barras
   )
```

Parâmetros

symbol

[in] Nome do instrumento financeiro, por exemplo, "EURUSD". Parâmetro sem nome obrigatório.

timeframe

[in] Período gráfico para o qual são solicitadas as barras. É definido pelo valor a partir da enumeração TIMEFRAME. Parâmetro sem nome obrigatório.

start_pos

[in] Número inicial da barra a partir da qual são solicitados os dados. A numeração das barras vai do presente para o passado, ou seja, a barra zero significa a atual. Parâmetro sem nome obrigatório.

count

[in] O número de barras que é necessário obter. Parâmetro sem nome obrigatório.

Valor retornado

Retorna barras na forma de array numpy contendo colunas nomeadas time, open, high, low, close, tick_volume, spread e real_volume. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

Para mais informações, veja a função CopyRates().

O terminal MetaTrader 5 fornece barras apenas nos limites do histórico disponível para o usuário nos gráficos. O número de barras disponíveis para o usuário é definido nas configurações com o parâmetro "Máx. de barras no gráfico"

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos o módulo pandas para exibir os dados recebidos na forma de uma tabela
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas
```

© 2000-2024, MetaQuotes Ltd.

```
pd.set_option('display.width', 1500)      # largura máxima da tabela

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# solicitamos 10 barras de GBPUSD D1 do dia atual
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

# exibimos cada elemento de dados recebidos numa nova linha
print("Exibimos os dados recebidos como estão")
for rate in rates:
    print(rate)

# a partir dos dados recebidos criamos o DataFrame
rates_frame = pd.DataFrame(rates)
# convertemos o tempo em segundos no formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# exibimos dados
print("\nExibimos o dataframe com dados")
print(rates_frame)

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Exibimos os dados como eles são
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

Exibimos o dataframe com dados
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
3 2020-02-18  1.30039  1.30486  1.29705  1.29952        62288       0            0
```

© 2000-2024, MetaQuotes Ltd.

4	2020-02-19	1.29952	1.30230	1.29075	1.29187	57909	0	0
5	2020-02-20	1.29186	1.29281	1.28489	1.28792	61033	0	0
6	2020-02-21	1.28802	1.29805	1.28746	1.29566	66386	0	0
7	2020-02-24	1.29426	1.29547	1.28865	1.29283	66933	0	0
8	2020-02-25	1.29290	1.30178	1.29142	1.30037	80121	0	0
9	2020-02-26	1.30036	1.30078	1.29136	1.29374	49286	0	0

Veja também

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_rates_range

Recebe barras a partir do terminal MetaTrader 5, no intervalo de datas especificado.

```
copy_rates_range(
   symbol,        // nome do símbolo
   timeframe,     // período gráfico
   date_from,     // data a partir da qual são solicitadas as barras
   date_to        // data para a qual são solicitadas as barras
   )
```

Parâmetros

symbol

[in] Nome do instrumento financeiro, por exemplo, "EURUSD". Parâmetro sem nome obrigatório.

timeframe

[in] Período gráfico para o qual são solicitadas as barras. É definido pelo valor a partir da enumeração TIMEFRAME. Parâmetro sem nome obrigatório.

date_from

[in] Data a partir da qual são solicitadas as barras. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. São usadas barras com tempo de abertura >= date_from. Parâmetro sem nome obrigatório.

date_to

[in] Data para a qual são solicitadas as barras. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. São usadas barras com tempo de abertura <= date_to. Parâmetro sem nome obrigatório.

Valor retornado

Retorna barras na forma de array numpy contendo colunas nomeadas time, open, high, low, close, tick_volume, spread e real_volume. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

Para mais informações, veja a função CopyRates().

O terminal MetaTrader 5 fornece barras apenas nos limites do histórico disponível para o usuário nos gráficos. O número de barras disponíveis para o usuário é definido nas configurações com o parâmetro "Máx. de barras no gráfico"

Python, ao criar um objeto datetime, usa o fuso horário local, enquanto o terminal MetaTrader 5 armazena o tempo de ticks e de abertura de barras no fuso horário UTC (sem deslocamento). Assim, para executar funções que usam tempo, deve-se criar um datetime no horário UTC. Os dados recebidos do terminal MetaTrader 5 têm hora UTC.

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
```

© 2000-2024, MetaQuotes Ltd.

```
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos o módulo pandas para exibir os dados recebidos na forma de uma tabela
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas exibidas
pd.set_option('display.width', 1500)      # max. largura para exibição da tabela
# importamos o módulo pytz para trabalhar com o fuso horário
import pytz

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# definimos o fuso horário como UTC
timezone = pytz.timezone("Etc/UTC")
# criamos o objeto datatime no fuso horário UTC para que não seja aplicado o deslocame
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# obtemos as barras com USDJPY M5 no intervalo 2020.01.10 00:00 - 2020.01.11 13:00 no
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
# exibimos cada elemento dos dados obtidos numa nova linha
print("Exibimos os dados obtidos como estão")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# criamos a partir dos dados obtidos DataFrame
rates_frame = pd.DataFrame(rates)
# convertemos o tempo em segundos no formato datetime
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# exibimos dados
print("\nExibimos o dataframe com dados")
print(rates_frame.head(10))
```

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Exibimos os dados como eles são
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
```

© 2000-2024, MetaQuotes Ltd.

```
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)

Exibimos o dataframe com dados
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-10 00:00:00  109.513  109.527  109.505  109.521           43       2
1 2020-01-10 00:05:00  109.521  109.549  109.518  109.543          215       8
2 2020-01-10 00:10:00  109.543  109.543  109.466  109.505           98      10
3 2020-01-10 00:15:00  109.504  109.534  109.502  109.517          155       8
4 2020-01-10 00:20:00  109.517  109.539  109.513  109.527           71       4
5 2020-01-10 00:25:00  109.526  109.537  109.484  109.520          106       9
6 2020-01-10 00:30:00  109.520  109.524  109.508  109.510          205       7
7 2020-01-10 00:35:00  109.510  109.510  109.491  109.496           44       8
8 2020-01-10 00:40:00  109.496  109.509  109.487  109.500           85       5
9 2020-01-10 00:45:00  109.500  109.504  109.487  109.489           82       7
```

Veja também

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_ticks_from

Recebe ticks do terminal MetaTrader 5, a partir da data especificada.

```
copy_ticks_from(
   symbol,        // nome do símbolo
   date_from,     // data a partir da qual são solicitados os ticks
   count,         // número de ticks solicitados
   flags          // combinação de sinalizadores que definem o tipo de ticks solicitado
   )
```

Parâmetros

symbol

[in] Nome do instrumento financeiro, por exemplo, "EURUSD". Parâmetro sem nome obrigatório.

date_from

[in] Data a partir da qual são solicitados os ticks. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

count

[in] Número de ticks que é necessário obter. Parâmetro sem nome obrigatório.

flags

[in] sinalizador que especifica o tipo de ticks solicitados. COPY_TICKS_INFO - ticks chamados pelas alterações do Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações em Last e Volume, COPY_TICKS_ALL - todos os ticks. Valores dos sinalizadores descritos na enumeração COPY_TICKS. Parâmetro sem nome obrigatório.

Valor retornado

Retorna os ticks na forma de array numpy contendo colunas nomeadas time, bid, ask, last e flags. O elemento flags pode ser uma combinação de sinalizadores a partir da enumeração TICK_FLAG. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

Para mais informações, veja a função CopyTicks.

Python, ao criar um objeto datetime, usa o fuso horário local, enquanto o terminal MetaTrader 5 armazena o tempo de ticks e de abertura de barras no fuso horário UTC (sem deslocamento). Assim, para executar funções que usam tempo, deve-se criar um datetime no horário UTC. Os dados recebidos do terminal MetaTrader 5 têm hora UTC.

COPY_TICKS define os tipos de ticks que podem ser solicitados com a ajuda das funções copy_ticks_from() e copy_ticks_range().

Identificador	Descrição
COPY_TICKS_ALL	todos os ticks

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
COPY_TICKS_INFO	ticks contendo alterações de preços Bid e/ou Ask
COPY_TICKS_TRADE	ticks contendo alterações do preço Last e/ou do volume (Volume)

TICK_FLAG define possíveis sinalizadores para ticks. Esses sinalizadores são usados para descrever os ticks recebidos pelas funções copy_ticks_from() e copy_ticks_range().

Identificador	Descrição
TICK_FLAG_BID	preço Bid alterado
TICK_FLAG_ASK	preço Ask alterado
TICK_FLAG_LAST	preço Last alterado
TICK_FLAG_VOLUME	volume alterado (Volume)
TICK_FLAG_BUY	preço da última compra alterado (Buy)
TICK_FLAG_SELL	preço da última venda alterado (Sell)

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# importamos o módulo pandas para exibir os dados recebidos na forma de uma tabela
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas
pd.set_option('display.width', 1500)      # largura máxima da tabela
# importamos o módulo pytz para trabalhar com o fuso horário
import pytz

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# definimos o fuso horário como UTC
timezone = pytz.timezone("Etc/UTC")
# criamos o objeto datetime no fuso horário UTC para que não seja aplicado o deslocame
# solicitamos 100 000 ticks de EURUSD a partir de 04/01/2019 no fuso horário UTC
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Ticks recebidos:",len(ticks))

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

# exibimos cada tick numa nova linha
print("Exibimos os ticks recebidos como são")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# a partir dos dados recebidos criamos o DataFrame
ticks_frame = pd.DataFrame(ticks)
# convertemos o tempo em segundos no formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# exibimos dados
print("\nExibimos o dataframe com ticks")
print(ticks_frame.head(10))
```

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks recebidos: 100 000
Exibimos os ticks recebidos como são
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

Exibimos o dataframe com ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134          (
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          (
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          (
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          (
```

© 2000-2024, MetaQuotes Ltd.

```
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          (
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          (
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          (
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          (
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          (
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          (
```

Veja também

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

copy_ticks_range

Recebe ticks a partir do terminal MetaTrader 5, no intervalo de datas especificado.

```
copy_ticks_range(
   symbol,        // nome do símbolo
   date_from,     // data a partir da qual são solicitados os ticks
   date_to,       // data segundo a qual são solicitados os ticks
   flags          // combinação de sinalizadores que definem o tipo de ticks solicitado
   )
```

Parâmetros

symbol

[in] Nome do instrumento financeiro, por exemplo, "EURUSD". Parâmetro sem nome obrigatório.

date_from

[in] Data a partir da qual são solicitados os ticks. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

date_to

[in] Data para a qual são solicitados os ticks. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

flags

[in] sinalizador que especifica o tipo de ticks solicitados. COPY_TICKS_INFO - ticks chamados pelas alterações do Bid e/ou Ask, COPY_TICKS_TRADE - ticks com alterações em Last e Volume, COPY_TICKS_ALL - todos os ticks. Valores dos sinalizadores descritos na enumeração COPY_TICKS. Parâmetro sem nome obrigatório.

Valor retornado

Retorna os ticks na forma de array numpy contendo colunas nomeadas time, bid, ask, last e flags. O elemento flags pode ser uma combinação de sinalizadores a partir da enumeração TICK_FLAG. Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

Para mais informações, veja a função CopyTicks.

Python, ao criar um objeto datetime, usa o fuso horário local, enquanto o terminal MetaTrader 5 armazena o tempo de ticks e de abertura de barras no fuso horário UTC (sem deslocamento). Assim, para executar funções que usam tempo, deve-se criar um datetime no horário UTC. Os dados recebidos do terminal MetaTrader 5 têm hora UTC, mas quando se tenta imprimi-los, o Python aplica novamente o deslocamento para a hora local. Por essa razão, a apresentação visual dos dados recebidos também precisa ser corrigida.

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
```

© 2000-2024, MetaQuotes Ltd.

```
print("MetaTrader5 package version: ",mt5.__version__)

# importamos o módulo pandas para exibir os dados recebidos na forma de uma tabela
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas exibidas
pd.set_option('display.width', 1500)      # max. largura para exibição da tabela
# importamos o módulo pytz para trabalhar com o fuso horário
import pytz

# estabelecemos conexão ao terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# definimos o fuso horário como UTC
timezone = pytz.timezone("Etc/UTC")
# criamos o objeto datatime no fuso horário UTC para que não seja aplicado o deslocame
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# solicitamos ticks de AUDUSD no intervalo de 11.01.2020 - 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Ticks recebidos:",len(ticks))

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
# exibimos cada tick numa nova linha
print("Exibimos os ticks recebidos como estão)
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# a partir dos dados recebidos criamos o DataFrame
ticks_frame = pd.DataFrame(ticks)
# convertemos o tempo em segundos no formato datetime
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# exibimos dados
print("\nExibimos o dataframe com ticks")
print(ticks_frame.head(10))
```

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Ticks recebidos: 37008
Exibimos os ticks recebidos como são
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
```

© 2000-2024, MetaQuotes Ltd.

```
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

Exibimos o dataframe com ticks
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134          (
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          (
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          (
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          (
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          (
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          (
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          (
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          (
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          (
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          (
```

Veja também

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2024, MetaQuotes Ltd.

orders_total

Obtém o número de ordens ativas.

```
orders_total()
```

Valor retornado

Valor inteiro.

Observação

A função é semelhante a OrdersTotal.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# verificamos a presença de ordens ativas
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("Orders not found")

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

orders_get, positions_total

© 2000-2024, MetaQuotes Ltd.

orders_get

Obtém ordens ativas com a capacidade de filtrar por símbolo ou bilhete. Existem 3 variantes de chamada.

Chamada sem parâmetros. Retorna as ordens ativas de todos os símbolos.

```
orders_get()
```

Uma chamada indicando o símbolo pelo qual é necessário receber as ordens ativas.

```
orders_get(
   symbol="SYMBOL"      // nome do símbolo
)
```

Uma chamada indicando o grupo de símbolos para o qual é necessário receber ordens ativas.

```
orders_get(
   group="GROUP"        // filtro para selecionar ordens por símbolos
)
```

Uma chamada indicando o bilhete da ordem.

```
orders_get(
   ticket=TICKET        // bilhete
)
```

symbol="SYMBOL"

[in] Nome do símbolo. Parâmetro nomeado opcional. Se o símbolo for especificado, o parâmetro *ticket* será ignorado.

group="GROUP"

[in] Filtro para selecionar apenas o grupo de símbolos necessários. Parâmetro nomeado opcional. Se um grupo for especificado, a função retornará apenas as ordens que satisfazem os critérios especificados para o nome do símbolo.

ticket=TICKET

[in] Bilhete da ordem (ORDER_TICKET). Parâmetro nomeado opcional.

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função permite de uma chamada receber todas as ordens ativas, a ligação OrdersTotal e OrderSelect é um análogo.

O parâmetro *group* permite filtrar ordens por símbolos, é permitido usar '*' no início e no final da linha.

O parâmetro *group* pode conter várias condições, separadas por vírgulas. A condição pode ser especificada como uma máscara usando '*'. Para exclusões, pode-se usar o símbolo de negação lógica '!'. Neste caso, todas as condições são aplicadas sequencialmente, ou seja, primeiro deve-se especificar as condições para inclusão no grupo e, em seguida, a condição de exclusão. Por

© 2000-2024, MetaQuotes Ltd.

exemplo, *group*="*, !EUR" significa que primeiro é necessário selecionar as ordens de todos os símbolos e, em seguida, excluir as que contêm o símbolo "EUR" no nome.

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas mostradas
pd.set_option('display.width', 1500)      # max. largura máxima da tabela exibida
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# exibimos informações sobre ordens ativas do símbolo GBPUSD
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("No orders on GBPUSD, error code={}".format(mt5.last_error()))
else:
    print("Total orders on GBPUSD:",len(orders))
    # exibimos todas as ordens ativas
    for order in orders:
        print(order)
print()

# obtemos uma lista de ordens com base em símbolos cujos nomes contenham "*GBP*"
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("No orders with group=\"*GBP*\", error code={}".format(mt5.last_error()))
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # exibimos essas posições como uma tabela usando pandas.DataFrame
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id', 'reason',
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Total orders on GBPUSD: 2
```

© 2000-2024, MetaQuotes Ltd.

```
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718, time
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419, time

orders_get(group="*GBP*")=4
      ticket          time_setup  time_setup_msc  time_expiration  type  type_time  ty
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
```

Veja também

orders_total, positions_get

© 2000-2024, MetaQuotes Ltd.

order_calc_margin

Retorna o tamanho da margem na moeda da conta para a operação de negociação especificada.

```
order_calc_margin(
   action,      // tipo de ordem (ORDER_TYPE_BUY ou ORDER_TYPE_SELL)
   symbol,      // nome do símbolo
   volume,      // volume
   price        // preço de abertura
   )
```

Parâmetros

action

[in] Tipo de ordem, sendo que pode assumir os valores da enumeração ORDER_TYPE. Parâmetro sem nome obrigatório.

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

volume

[in] Volume da operação de negociação. Parâmetro sem nome obrigatório.

price

[in] Preço de abertura. Parâmetro sem nome obrigatório.

Valor retornado

Valor real em caso de execução bem-sucedida, caso contrário, None. Informações de erro podem ser obtidas usando last_error().

Observação

A função permite estimar a margem necessária para o tipo de ordem especificado na conta corrente e no ambiente de mercado atual, excluindo os pedidos pendentes atuais e as posições abertas. É semelhante a OrderCalcMargin.

ORDER_TYPE

Identificador	Descrição
ORDER_TYPE_BUY	Ordem de mercado para compra
ORDER_TYPE_SELL	Ordem de mercado para venda
ORDER_TYPE_BUY_LIMIT	Ordem pendente Buy Limit
ORDER_TYPE_SELL_LIMIT	Ordem pendente Sell Limit
ORDER_TYPE_BUY_STOP	Ordem pendente Buy Stop
ORDER_TYPE_SELL_STOP	Ordem pendente Sell Stop
ORDER_TYPE_BUY_STOP_LIMIT	Ao atingir o preço da ordem, uma ordem Buy Limit é colocada segundo o preço Stop Limit

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
ORDER_TYPE_SELL_STOP_LIMIT	Ao atingir o preço da ordem, é colocada uma ordem pendente Sell Limit de acordo com o preço StopLimit
ORDER_TYPE_CLOSE_BY	Ordem de fechamento da posição oposta

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos a moeda da conta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# geramos uma lista de símbolos
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Symbols to check margin:", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lot margin: {} {}".format(symbol,lot,margin,account_curren
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD

Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY', 'GBPJPY')
```

© 2000-2024, MetaQuotes Ltd.

```
EURUSD buy 0.1 lot margin: 109.91 USD
GBPUSD buy 0.1 lot margin: 122.73 USD
USDJPY buy 0.1 lot margin: 100.0 USD
USDCHF buy 0.1 lot margin: 100.0 USD
EURJPY buy 0.1 lot margin: 109.91 USD
GBPJPY buy 0.1 lot margin: 122.73 USD
```

Veja também

order_calc_profit, order_check

© 2000-2024, MetaQuotes Ltd.

order_calc_profit

Retorna o valor do lucro na moeda da conta para a operação de negociação especificada.

```
order_calc_profit(
   action,          // tipo de ordem (ORDER_TYPE_BUY ou ORDER_TYPE_SELL)
   symbol,          // nome do símbolo
   volume,          // volume
   price_open,      // preço de abertura
   price_close      // preço de fechamento
   );
```

Parâmetros

action

[in] Tipo de ordem, sendo que pode assumir um dos dois valores da enumeração ORDER_TYPE: ORDER_TYPE_BUY ou ORDER_TYPE_SELL. Parâmetro sem nome obrigatório.

symbol

[in] Nome do instrumento financeiro. Parâmetro sem nome obrigatório.

volume

[in] Volume da operação de negociação. Parâmetro sem nome obrigatório.

price_open

[in] Preço de abertura. Parâmetro sem nome obrigatório.

price_close

[in] Preço de fechamento. Parâmetro sem nome obrigatório.

Valor retornado

Valor real em caso de execução bem-sucedida, caso contrário, None. Informações de erro podem ser obtidas usando last_error().

Observação

A função permite avaliar o resultado de uma operação de negociação na conta corrente e no ambiente de mercado atual. É semelhante a OrderCalcProfit.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# obtemos a moeda da conta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# fazemos uma lista de símbolos
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symbols to check margin:", symbols)
# avaliamos valores de lucro para compras e vendas
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}) failed, skipped",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*po
    if buy_profit!=None:
        print("   buy {} {} lot: profit on {} points => {} {}".format(symbol,lot,dista
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) failed, error code =",mt5.last_error
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots: profit on {} points => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) failed, error code =",mt5.last_error
    print()

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

```
Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
Symbols to check margin: ('EURUSD', 'GBPUSD', 'USDJPY')
   buy EURUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell EURUSD 1.0 lot: profit on 300 points => 300.0 USD

   buy GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
```

© 2000-2024, MetaQuotes Ltd.

```
buy USDJPY 1.0 lot: profit on 300 points => 276.54 USD
sell USDJPY 1.0 lot: profit on 300 points => 278.09 USD
```

Veja também

order_calc_margin, order_check

© 2000-2024, MetaQuotes Ltd.

order_check

Verifica que há fundos suficientes para realizar a operação de negociação requerida. Os resultados da verificação são retornados como estruturas MqlTradeCheckResult.

```
order_check(
   request      // estrutura da solicitação
   );
```

Parâmetros

request

[in] Estrutura do tipo MqlTradeRequest que descreve a ação de negociação requerida. Parâmetro sem nome obrigatório. Veja abaixo um exemplo de preenchimento de solicitação e a composição de enumerações.

Valor retornado

Resultado de verificação como estrutura MqlTradeCheckResult. O campo *request* na resposta contém a estrutura da solicitação de negociação transferida para order_check().

Observação

Uma verificação bem-sucedida **não é um sinal de que a operação de negociação solicitada certamente seja executada com êxito**. A função order_check é semelhante a OrderCheck.

TRADE_REQUEST_ACTIONS

Identificador	Descrição
TRADE_ACTION_DEAL	Definir uma ordem de negociação para executar imediatamente uma transação com os parâmetros especificados (colocar uma ordem a mercado)
TRADE_ACTION_PENDING	Definir uma ordem de negociação para executar uma transação nas condições especificados (colocar uma ordem pendente)
TRADE_ACTION_SLTP	Alterar os valores de Stop Loss e de Take Profit da posição aberta
TRADE_ACTION_MODIFY	Alterar os parâmetros de uma ordem de negociação colocada anteriormente
TRADE_ACTION_REMOVE	Excluir uma ordem pendente colocada anteriormente
TRADE_ACTION_CLOSE_BY	Fechar a posição oposta

ORDER_TYPE_FILLING

Identificador	Descrição
ORDER_FILLING_FOK	Esta política de execução significa que a ordem pode ser executada exclusivamente com base no volume especificado. Se, no mercado, nesse momento, não estiver presente um volume suficiente de instrumento financeiro, a

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
	ordem não será executada. O volume necessário pode ser composto pelas variadas ofertas atualmente disponíveis no mercado.
ORDER_FILLING_IOC	Significa consentimento para concluir uma transação de acordo com o valor máximo disponível no mercado dentro dos limites especificados na ordem. No caso de impossibilidade de total execução, a ordem será executada no volume disponível, e o volume não disponível será cancelado.
ORDER_FILLING_RETURN	Este modo é usada para ordenes a mercado (ORDER_TYPE_BUY e ORDER_TYPE_SELL), ordens limitadas e ordens stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT) e apenas nos modos "Execução a mercado" e "Execução de bolsa". No caso de uma execução parcial, a ordem de mercado ou limit com um volume residual não é retirada e continua a vigorar. Durante a ativação das ordens ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT, será criado a ordem limite correspondente ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT com execução ORDER_FILLING_RETURN.

ORDER_TYPE_TIME

Identificador	Descrição
ORDER_TIME_GTC	A ordem permanecerá na fila até ser retirada
ORDER_TIME_DAY	A ordem será válida apenas durante o dia de negociação atual
ORDER_TIME_SPECIFIED	A ordem será válida até a data de vencimento
ORDER_TIME_SPECIFIED_DAY	A ordem será válida até 23:59:59 do dia especificado. Se essa hora não coincidir com a sessão de trading, o vencimento acontecerá ao iniciar a próxima negociação.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao terminal MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

© 2000-2024, MetaQuotes Ltd.

```
    quit()

# obtemos a moeda da conta
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# preparamos a estrutura da solicitação
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# se o símbolo não estiver disponível no MarketWatch, adicionamo-lo
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# preparamos a solicitação
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# verificamos e exibimos o resultado como está
result = mt5.order_check(request)
print(result);
# solicitamos o resultado na forma de um dicionário e exibimos elemento por elemento
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # se esta for uma estrutura de uma solicitação de negociação, também a exibiremos
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict
```

```
# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
   retcode=0
   balance=101300.53
   equity=68319.53
   profit=-32981.0
   margin=51193.67
   margin_free=17125.86
   margin_level=133.45308121101692
   comment=Done
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=1.0,
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
       traderequest: symbol=USDJPY
       traderequest: volume=1.0
       traderequest: price=108.081
       traderequest: stoplimit=0.0
       traderequest: sl=107.98100000000001
       traderequest: tp=108.181
       traderequest: deviation=10
       traderequest: type=0
       traderequest: type_filling=2
       traderequest: type_time=0
       traderequest: expiration=0
       traderequest: comment=python script
       traderequest: position=0
       traderequest: position_by=0
```

Veja também

order_send, OrderCheck, Tipos de operações de negociação, Estrutura da solicitação de negociação, Estrutura dos resultados de verificação da solicitação de negociação, Estrutura do resultado da solicitação de negociação

© 2000-2024, MetaQuotes Ltd.

order_send

Envia do terminal para o servidor de negociação uma solicitação para concluir uma operação de negociação. A função é semelhante a OrderSend.

```
order_send(
   request      // estrutura da solicitação
   );
```

Parâmetros

request

[in] Estrutura do tipo MqlTradeRequest que descreve a ação de negociação requerida. Parâmetro sem nome obrigatório. Veja abaixo um exemplo de preenchimento de solicitação e a composição de enumerações.

Valor retornado

Resultado da execução como uma estrutura MqlTradeResult. O campo *request* na resposta contém a estrutura da solicitação de negociação transferida para order_send().

Estrutura da solicitação de negociação MqlTradeRequest

Campo	Descrição
action	Tipo de operação de negociação. O valor pode ser um dos valores da enumeração TRADE_REQUEST_ACTIONS
magic	ID do EA. Permite realizar o processamento analítico de ordens de negociação. Cada EA pode definir seu próprio identificador exclusivo ao enviar uma solicitação de negociação.
order	Bilhete da ordem. Necessário para modificar ordens pendentes
symbol	Nome do instrumento de negociação para o qual é colocada a ordem. Não é necessário para operações de modificação de ordens e fechamento de posições
volume	Volume de transação solicitado em lotes. O valor real do volume ao abrir uma transação dependerá do tipo de ordem segundo execução.
price	O preço que, ao ser atingido, executa uma ordem pendente. Para ordens a mercado de instrumentos com o tipo de execução"Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET), que têm o tipo TRADE_ACTION_DEAL, não é necessária nenhuma indicação do preço
stoplimit	Preço segundo o qual será colocada a ordem pendente Limit quando o preço atingir o valor price (esta condição é obrigatória). Até este momento, a ordem pendente não é exibida no sistema de negociação.
sl	Preço segundo o qual será colocada a ordem Stop Loss quando o preço se mover numa direção desfavorável
tp	Preço segundo o qual será colocada a ordem Take Profit quando o preço se mover numa direção favorável

© 2000-2024, MetaQuotes Ltd.

Campo	Descrição
deviation	Desvio máximo aceitável em relação ao preço solicitado, definido em pontos
type	Tipo de ordem. O valor pode ser um dos valores da enumeração ORDER_TYPE
type_filling	Tipo de ordem segundo sua execução. O valor pode ser um dos valores da enumeração ORDER_TYPE_FILLING
type_time	Tipo de pedido segundo expiração. O valor pode ser um dos valores da enumeração ORDER_TYPE_TIME
expiration	Prazo de validade da ordem (para ordens do tipo TIME_SPECIFIED)
comment	Comentário da ordem
position	Bilhete da posição. Deve ser preenchido ao alterar e fechar a posição para a sua identificação inequívoca. Normalmente, corresponde ao bilhete de ordem, do qual, como resultado, a posição foi aberta.
position_by	Bilhete da posição oposta. Utiliza-se ao fechar uma posição usando outra oposta, se estiver aberta no mesmo instrumento, mas na direção oposta.

Observação

Uma solicitação de negociação passa por vários estágios de verificação no servidor de negociação. Primeiro, verifica-se que o preenchimento de todos os campos obrigatórios da solicitação *request* está correto e, se não houver erros, o servidor aceitará a ordem para continuar a processá-la. Para obter informações completas sobre a execução de operações de negociação, consulte a descrição da função OrderSend.

Exemplo:

```
import time
import MetaTrader5 as mt5

# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# preparamos a estrutura de solicitação para compra
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()
```

© 2000-2024, MetaQuotes Ltd.

```
# se o símbolo não estiver disponível no MarketWatch, adicionamo-lo
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# enviamos a solicitação de negociação
result = mt5.order_send(request)
# verificamos o resultado da execução
print("1. order_send(): by {} {} lots at {} with deviation={} points".format(symbol,lo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send failed, retcode={}".format(result.retcode))
   # solicitamos o resultado na forma de dicionário e exibimos elemento por elemento
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        #se esta for uma estrutura de uma solicitação de negociação, também a exibiren
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_
    print("shutdown() and quit")
    mt5.shutdown()
    quit()

print("2. order_send done, ", result)
print("   opened position with POSITION_TICKET={}".format(result.order))
print("   sleep 2 seconds before closing position #{}".format(result.order))
```

© 2000-2024, MetaQuotes Ltd.

```
time.sleep(2)
# criamos uma solicitação de fechamento
position_id=result.order
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# enviamos a solicitação de negociação
result=mt5.order_send(request)
# verificamos o resultado da execução
print("3. close position #{}: sell {} {} lots at {} with deviation={} points".format(p
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send failed, retcode={}".format(result.retcode))
    print("   result",result)
else:
    print("4. position #{} closed, {}".format(position_id,result))
   # solicitamos o resultado na forma de dicionário e exibimos elemento por elemento
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        #se esta for uma estrutura de uma solicitação de negociação, também a exibirem
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

Resultado
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
1. order_send(): by USDJPY 0.1 lots at 108.023 with deviation=20 points
2. order_send done,  OrderSendResult(retcode=10009, deal=535084512, order=557416535, v
   opened position with POSITION_TICKET=557416535
   sleep 2 seconds before closing position #557416535
3. close position #557416535: sell USDJPY 0.1 lots at 108.018 with deviation=20 points
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631, order=55
```

© 2000-2024, MetaQuotes Ltd.

```
    retcode=10009
    deal=535084631
    order=557416654
    volume=0.1
    price=108.015
    bid=108.015
    ask=108.02
    comment=Request executed
    request_id=55
    retcode_external=0
    request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=0.1,
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=0.1
        traderequest: price=108.018
        traderequest: stoplimit=0.0
        traderequest: sl=0.0
        traderequest: tp=0.0
        traderequest: deviation=20
        traderequest: type=1
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script close
        traderequest: position=557416535
        traderequest: position_by=0
```

Veja também

order_check, OrderSend, Tipos de operações de negociação, Estrutura da solicitação de negociação, Estrutura dos resultados de verificação da solicitação de negociação, Estrutura do resultado da solicitação de negociação

© 2000-2024, MetaQuotes Ltd.

positions_total

Obtém o número de posições abertas.

```
positions_total()
```

Valor retornado

Valor inteiro.

Observação

A função é semelhante a PositionsTotal.

Exemplo:

```
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# verificamos a presença de posições abertas
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("Positions not found")

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

Veja também

positions_get, orders_total

© 2000-2024, MetaQuotes Ltd.

positions_get

Obtém posições abertas com a capacidade de filtrar por símbolo ou ticket. Existem 3 variantes de chamada.

Chamada sem parâmetros. Retorna posições abertas de todos os símbolos.

```
positions_get()
```

Uma chamada indicando o símbolo para o qual é necessário receber posições abertas.

```
positions_get(
   symbol="SYMBOL"      // nome do símbolo
)
```

Uma chamada indicando o grupo de símbolos para os quais é necessário receber posições abertas.

```
positions_get(
   group="GROUP"        // filtro para selecionar posições por símbolos
)
```

Uma chamada indicando o bilhete da posição.

```
positions_get(
   ticket=TICKET        // bilhete
)
```

Parâmetros

symbol="SYMBOL"

[in] Nome do símbolo. Parâmetro nomeado opcional. Se o símbolo for especificado, o parâmetro *ticket* será ignorado.

group="GROUP"

[in] Filtro para selecionar apenas o grupo de símbolos necessários. Parâmetro nomeado opcional. Se um grupo for especificado, a função retornará apenas as posições que satisfazem os critérios especificados para o nome do símbolo.

ticket=TICKET

[in] Bilhete da posição (POSITION_TICKET). Parâmetro nomeado opcional.

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função permite de uma chamada obter todas as posições abertas, a ligação PositionsTotal e PositionSelect é um análogo.

O parâmetro *group* pode conter várias condições, separadas por vírgulas. A condição pode ser especificada como uma máscara usando '*'. Para exclusões, pode-se usar o símbolo de negação lógica '!'. Neste caso, todas as condições são aplicadas sequencialmente, ou seja, primeiro deve-se especificar as condições para inclusão no grupo e, em seguida, a condição de exclusão. Por

© 2000-2024, MetaQuotes Ltd.

exemplo, *group*="*, !EUR" significa que primeiro é necessário selecionar as posições de todos os símbolos e, em seguida, excluir as que contêm o símbolo "EUR" no nome.

Exemplo:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas mostradas
pd.set_option('display.width', 1500)      # max. largura máxima da tabela exibida
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos as posições abertas com base no USDCHF
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("No positions on USDCHF, error code={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Total positions on USDCHF =",len(positions))
    # imprimimos todas as posições abertas
    for position in positions:
        print(position)

# obtemos uma lista de posições com base em símbolos cujos nomes contenham "*USD*"
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("No positions with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # exibimos essas posições como uma tabela usando pandas.DataFrame
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1, inp
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()


Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

positions_get(group="*USD*")=5
```

© 2000-2024, MetaQuotes Ltd.

	ticket	time	type	magic	identifier	reason	volume	price_open
0	548297723	2020-03-18 15:00:55	1	0	548297723	3	0.01	1.09301
1	548655158	2020-03-18 20:31:26	0	0	548655158	3	0.01	1.08676
2	548663803	2020-03-18 20:40:04	0	0	548663803	3	0.01	1.08640
3	548847168	2020-03-19 01:10:05	0	0	548847168	3	0.01	1.09545
4	548847194	2020-03-19 01:10:07	0	0	548847194	3	0.02	1.09536

Veja também

positions_total, orders_get

© 2000-2024, MetaQuotes Ltd.

history_orders_total

Obtém o número de ordens no histórico de negociação no intervalo especificado.

```
history_orders_total(
   date_from,     // data a partir da qual são solicitadas as ordens
   date_to        // data segundo a qual são solicitadas as ordens
   )
```

Parâmetros

date_from

[in] Data a partir da qual são solicitadas as ordens. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

date_to

[in] Data segundo a qual são solicitadas as ordens É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

Valor retornado

Valor inteiro.

Observação

A função é semelhante a HistoryOrdersTotal.

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos o número de ordens no histórico
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Total history orders=",history_orders)
else:
    print("Orders not found in history")

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

Veja também

history_orders_get, history_deals_total

© 2000-2024, MetaQuotes Ltd.

history_orders_get

Obtém ordens do histórico de negociação com a capacidade de filtrar por bilhete ou posição. Existem 3 variantes de chamada.

Uma chamada indicando o intervalo de tempo. Retorna todas as ordens dentro do intervalo especificado.

```
history_orders_get(
   date_from,           // data a partir da qual são solicitadas as ordens
   date_to,             // data segundo a qual são solicitados os ticks
   group="GROUP"        // filtro para selecionar ordens por símbolos
   )
```

Uma chamada indicando o bilhete da ordem. Retorna a ordem com o bilhete especificado.

```
history_orders_get(
   ticket=TICKET        // bilhete da ordem
)
```

Uma chamada indicando o bilhete da posição. Retorna todas as posições com a propriedade ORDER_POSITION_ID, com o bilhete da posição especificado.

```
history_orders_get(
   position=POSITION    // bilhete da posição
)
```

Parâmetros

date_from

[in] Data a partir da qual são solicitadas as ordens. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório, especificado primeiro.

date_to

[in] Data segundo a qual são solicitadas as ordens É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório, especificado em segundo lugar.

group="GROUP"

[in] Filtro para selecionar apenas o grupo de símbolos necessários. Parâmetro nomeado opcional. Se um grupo for especificado, a função retornará apenas as ordens do histórico que satisfazem os critérios especificados para o nome do símbolo.

ticket=TICKET

[in] Bilhete da ordem deve ser recebido. Parâmetro opcional. Se não for especificado o bilhete da ordem, não será aplicado o filtro.

position=POSITION

[in] Bilhete da posição (é armazenado em ORDER_POSITION_ID), para o qual é necessário obter todas as ordens. Parâmetro opcional. Se não for especificado o bilhete da posição, não será aplicado o filtro.

Valor retornado

© 2000-2024, MetaQuotes Ltd.

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função permite de uma chamada receber todas as ordens ativas, a ligação HistoryOrdersTotal e HistoryOrderSelect é um análogo.

O parâmetro *group* pode conter várias condições, separadas por vírgulas. A condição pode ser especificada como uma máscara usando '*'. Para exclusões, pode-se usar o símbolo de negação lógica '!'. Neste caso, todas as condições são aplicadas sequencialmente, ou seja, primeiro deve-se especificar as condições para inclusão no grupo e, em seguida, a condição de exclusão. Por exemplo, *group*="*, !EUR" significa que primeiro é necessário selecionar as transações de todos os símbolos e, em seguida, excluir as que contêm o símbolo "EUR" no nome.

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas mostradas
pd.set_option('display.width', 1500)      # max. largura máxima da tabela exibida
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos o número de ordens no histórico
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("No history orders with group=\"*GBP*\", error code={}".format(mt5.last_erro
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# exibimos todas as ordens históricas segundo o bilhete da posição
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("No orders with position #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total history orders on position #{}: {}".format(position_id,len(position_h
   # exibimos todas as ordens históricas que possuem o bilhete da posição especificado
    for position_order in position_history_orders:
```

```
        print(position_order)
    print()
    # exibimos essas posições como uma tabela usando pandas.DataFrame
    df=pd.DataFrame(list(position_history_orders),columns=position_history_orders[0].
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume_
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

Total history orders on position #530218319: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681, time
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265, time

      ticket          time_setup  time_setup_msc           time_done  time_done_msc  t
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
```

Veja também

history_deals_total, history_deals_get

© 2000-2024, MetaQuotes Ltd.

history_deals_total

Obtém o número de transações no histórico de negociação no intervalo especificado.

```
history_deals_total(
   date_from,       // data a partir da qual são solicitadas as transações
   date_to          // data segundo a qual são solicitadas as transações
   )
```

Parâmetros

date_from

[in] Data a partir da qual são solicitadas as transações. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

date_to

[in] Data segundo a qual são solicitadas as transações. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório.

Valor retornado

Valor inteiro.

Observação

A função é semelhante a HistoryDealsTotal.

Exemplo:

```
from datetime import datetime
import MetaTrader5 as mt5
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos o número de transações no histórico
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Total deals=",deals)
else:
    print("Deals not found in history")

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()
```

© 2000-2024, MetaQuotes Ltd.

Veja também

history_deals_get, history_orders_total

© 2000-2024, MetaQuotes Ltd.

history_deals_get

Obtém transações do histórico de negociação no intervalo especificado, com a capacidade de filtrar por bilhete ou posição.

Uma chamada indicando o intervalo de tempo. Retorna todas as transações dentro do intervalo especificado.

```
history_deals_get(
   date_from,           // data a partir da qual são solicitadas as transações
   date_to,             // data segundo a qual são solicitadas as transações
   group="GROUP"        // filtro para selecionar ordens por símbolos
   )
```

Uma chamada indicando o bilhete da ordem. Retorna todas as transações que possuem o bilhete de ordem especificado na propriedade DEAL_ORDER.

```
history_deals_get(
   ticket=TICKET        // bilhete da ordem
)
```

Uma chamada indicando o bilhete da posição. Retorna todas as transações que possuem o bilhete de posição especificado na DEAL_POSITION_ID.

```
history_deals_get(
   position=POSITION    // bilhete da posição
)
```

Parâmetros

date_from

[in] Data a partir da qual são solicitadas as ordens. É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório, especificado primeiro.

date_to

[in] Data segundo a qual são solicitadas as ordens É definido pelo objeto datetime ou pelo número de segundos que passaram desde 1970.01.01. Parâmetro sem nome obrigatório, especificado em segundo lugar.

group="GROUP"

[in] Filtro para selecionar apenas o grupo de símbolos necessários. Parâmetro nomeado opcional. Se um grupo for especificado, a função retornará apenas as transações que satisfazem os critérios especificados para o nome do símbolo.

ticket=TICKET

[in] Bilhete da ordem (é armazenado em DEAL_ORDER), para o qual é necessário obter todas as transações. Parâmetro opcional. Se não for especificado o bilhete da ordem, não será aplicado o filtro.

position=POSITION

[in] Bilhete da posição (é armazenado em DEAL_POSITION_ID), para o qual é necessário obter todas as ordens. Parâmetro opcional. Se não for especificado o bilhete da posição, não será aplicado o filtro.

© 2000-2024, MetaQuotes Ltd.

Valor retornado

Retorna informações na forma de uma estrutura de tuplas nomeadas (namedtuple). Em caso de erro, retorna None, enquanto informações sobre o erro podem ser obtidas usando last_error().

Observação

A função permite de uma chamada receber todas as transações do histórico para o período especificado, a ligação HistoryOrdersTotal e HistoryOrderSelect é um análogo.

O parâmetro *group* permite filtrar transações por símbolos, é permitido usar '*' no início e no final da linha.

O parâmetro *group* pode conter várias condições, separadas por vírgulas. A condição pode ser especificada como uma máscara usando '*'. Para exclusões, pode-se usar o símbolo de negação lógica '!'. Neste caso, todas as condições são aplicadas sequencialmente, ou seja, primeiro deve-se especificar as condições para inclusão no grupo e, em seguida, a condição de exclusão. Por exemplo, *group*="*, !EUR" significa que primeiro é necessário selecionar as transações de todos os símbolos e, em seguida, excluir as que contêm o símbolo "EUR" no nome.

Exemplo:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # número de colunas mostradas
pd.set_option('display.width', 1500)      # max. largura máxima da tabela exibida
# exibimos dados sobre o pacote MetaTrader5
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# estabelecemos a conexão ao MetaTrader 5
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# obtemos o número de transações no histórico
from_date=datetime(2020,1,1)
to_date=datetime.now()
# obtemos no intervalo especificado as transações com base em símbolos cujos nomes cor
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("No deals with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,ler

# obtemos transações cujos símbolos não contêm "EUR" nem "GBP"
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("No deals, error code={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(de
```

© 2000-2024, MetaQuotes Ltd.

```
    # exibimos todas as transações recebidos como estão
    for deal in deals:
        print("  ",deal)
    print()
    # exibimos essas transações como uma tabela usando pandas.DataFrame
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# obtemos todas as transações que pertencem à posição #530218319
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("No deals with position #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals with position id #{}: {}".format(position_id, len(position_deals)))
    # exibimos essas transações como uma tabela usando pandas.DataFrame
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# concluímos a conexão ao terminal MetaTrader 5
mt5.shutdown()

Resultado:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419, type=
   TradeDeal(ticket=507962919, order=530218319, time=1582303777, time_msc=158230377758
   TradeDeal(ticket=513149059, order=535548147, time=1583176242, time_msc=158317624226
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=158351000389
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=158351002505
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=158352020122
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=158352020297

      ticket      order                time       time_msc  type  entry  magic  posit
0  506966741          0 2020-02-20 12:35:25  1582202125419     2      0      0
1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    530
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    530
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    539
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    539
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    539
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    539
```

© 2000-2024, MetaQuotes Ltd.

```
Deals with position id #530218319: 2
      ticket      order                time       time_msc  type  entry  magic  positi
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
```

Veja também

history_deals_total, history_orders_get

© 2000-2024, MetaQuotes Ltd.

Modelos ONNX em aprendizado de máquina

O ONNX (Open Neural Network Exchange) é uma biblioteca aberta para a criação de redes neurais de aprendizagem profunda. Este projeto tem várias vantagens importantes:

- Suporte a ONNX que é oferecido por grandes empresas, como Microsoft, Facebook, Amazon e outros parceiros.
- O formato aberto permite a implementação de conversão de formatos entre diferentes ferramentas de aprendizado de máquina, ONNXMLTools da Microsoft oferece a possibilidade de converter modelos no formato ONNX.
- MQL5 oferece autoconversão de dados de entrada e saída do modelo se o tipo de parâmetro passado não corresponder ao modelo.
- Criação de modelos ONNX que pode ser realizada com muitas ferramentas de aprendizado de máquina, atualmente suportadas pelo Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch e OpenCV; também há interfaces para muitas outras estruturas e bibliotecas populares.
- A linguagem MQL5 possibilita implementar o modelo ONNX em uma estratégia de negociação e tirar o máximo proveito da plataforma MetaTrader 5 para trabalhar nos mercados financeiros de maneira eficiente.
- Antes de executar o modelo na negociação real, você pode verificar o modelo em relação aos dados históricos no testador de estratégias sem o uso de ferramentas de terceiros.

As seguintes funções são disponibilizadas na MQL5 para trabalhar com o ONNX:

Função	Ação
OnnxCreate	Cria uma sessão ONNX com carregamento de modelo a partir de um arquivo *.onnx
OnnxCreateFromBuffer	Cria uma sessão ONNX com carregamento de modelo a partir de uma matriz de dados
OnnxRelease	Exclusão de sessão ONNX
OnnxRun	Execução do modelo ONNX
OnnxGetInputCount	Obtenção do número de parâmetros de entrada do modelo ONNX
OnnxGetOutputCount	Obtenção do número de parâmetros de saída do modelo ONNX
OnnxGetInputName	Obtenção do nome do parâmetro de entrada do modelo por índice
OnnxGetOutputName	Obtenção do nome do parâmetro de saída do modelo por índice
OnnxGetInputTypeInfo	Obtenção de uma descrição do tipo de parâmetro de entrada do modelo
OnnxGetOutputTypeInfo	Obtenção de uma descrição do tipo de parâmetro de saída do modelo
OnnxSetInputShape	Define a dimensionalidade dos dados de entrada do modelo por índice
OnnxSetOutputShape	Define a dimensionalidade dos dados de saída do modelo por índice

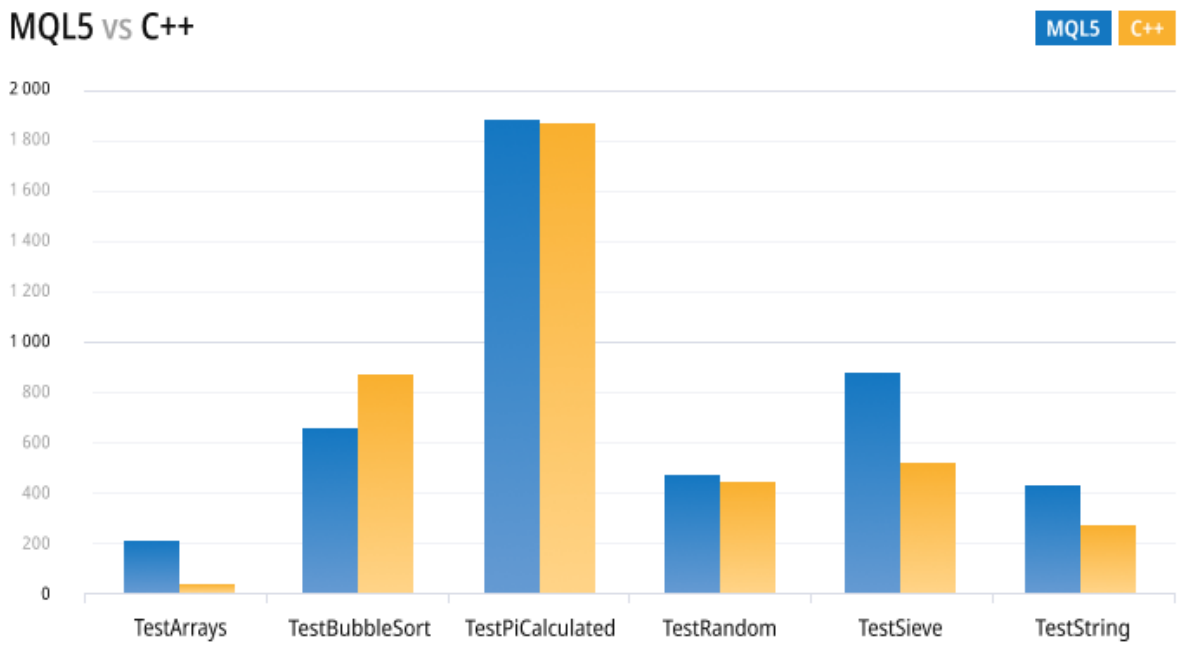
© 2000-2024, MetaQuotes Ltd.

Suporte a ONNX em MQL5

ONNX é um formato aberto criado para representar modelos de aprendizado de máquina. Esse padrão define um conjunto comum de operadores e formatos de arquivo que permite que os desenvolvedores usem os modelos em diferentes plataformas e ambientes de execução, com diferentes ferramentas e compiladores.

Esse formato aberto ONNX permite obter e mover modelos de aprendizado de máquina entre diferentes plataformas e ferramentas de aprendizado de máquina. É por isso que o suporte a ONNX foi adicionado à linguagem MQL5, assim os desenvolvedores de IA podem executar modelos criados em um ambiente de execução de alto desempenho na plataforma MetaTrader 5.

A linguagem MQL5 é tão rápida quanto os aplicativos desenvolvidos em C++. Os resultados da execução de testes padrão em MQL5 e C++ confirmam esse fato. Quanto menor a coluna, menos tempo é gasto para a execução, melhor o resultado (tempo em milissegundos). Testes realizados no Windows 10 (build 17763) x64, Xeon E5-2630 v4 @ 2.20GHz, Memória: 65457 Mb.



Com as novas operações de negociação assíncronas e o suporte nativo ao ONNX, surgem novas possibilidades que antes estavam disponíveis apenas para profissionais selecionados e operadores institucionais. O suporte a ONNX em MQL5 permite que os operadores treinem modelos para mercados financeiros em seu ambiente de desenvolvimento preferido e, por sua vez, negociem com baixos custos de rede, com uma alta taxa de atualização de cotações e com o envio assíncrono de ordens.

Atualmente, o ONNX está sendo desenvolvido e suportado em conjunto pela Microsoft, Facebook, Amazon e outros parceiros, garantindo o desenvolvimento desse projeto de código aberto.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Conversão de formatos

O formato ONNX é aberto, permitindo que modelos de várias ferramentas de aprendizado de máquina sejam armazenados nele. Esse formato é suportado por muitas plataformas, incluindo Chainer, Caffee2 e PyTorch.

Uma das ferramentas mais populares para converter modelos no formato ONNX é o ONNXMLTools desenvolvida pela Microsoft.

As instruções para instalar e usar o ONNXMLTools estão disponíveis no GitHub. Atualmente, há suporte para os seguintes pacotes de ferramentas:

- Keras (shell do conversor keras2onnx)
- Tensorflow (shell do conversor tf2onnx)
- scikit-learn (shell do conversor skl2onnx)
- Apple Core ML
- Spark ML (modo experimental)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

A instalação do ONNXMLTools é simples e está descrita na página do projeto em https://github.com/onnx/onnxmltools#install, onde há também alguns exemplos de conversão de modelos.onde há também alguns exemplos de conversões de modelos

© 2000-2024, MetaQuotes Ltd.

Autoconversão de valores de entrada e saída ao executar modelos ONNX

Na versão atual do ONNX em MQL5 como valores do tipo input/output são suportados apenas tensores, que são matrizes de dados com um dos seguintes tipos de elementos:

Tipo ONNX	Em conformidade com o tipo MQL5
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

Como valores do tipo input/output no modelo ONNX só é possível transferir matrizes, vetores ou matrizes (**Dados**).

Se o tipo de parâmetro passado não corresponder ao tipo de parâmetro do modelo ONNX e se o sinalizador OnnxRun não for especificado ao chamar ONNX_NO_CONVERSION, será usada a autoconversão. A autoconversão implica que, antes de executar o modelo ONNX, os dados do usuário **Dados**serão copiados para os sensores ONNX com conversão.

Ao executar um modelo ONNX sem autoconversão, o modelo ONNX é calculado em **Dados** sem nenhuma cópia adicional.

> **ADVERTÊNCIA!**A autoconversão não controla o truncamento, portanto, deve-se tomar cuidado com os dados e tipos de dados que são passados para o modelo ONNX.

A autoconversão é compatível com os seguintes tipos de ONNX:

© 2000-2024, MetaQuotes Ltd.

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8
- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

Tipos não suportados:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Regras de autoconversão por tipos de tensor

Se o tipo MQL5 não estiver entre os tipos suportados pelo modelo, o modelo ONNX será encerrado com o erro ERR_ONNX_NOT_SUPPORTED (código de erro 5802).

Nota: durante a autoconversão, o tipo color é processado como uint e o tipo datetime é processado como long.

Autoconversão de valores input

Tipo ONNX (tipo de elemento tensor)	Tipo MQL5, suportado pela autoconversão
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Ao converter elementos de **Dados** é verificado por uma comparação simples de zero a zero.
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Veja. ONNX_DATA_TYPE_FLOAT

© 2000-2024, MetaQuotes Ltd.

Tipo ONNX (tipo de elemento tensor)	Tipo MQL5, suportado pela autoconversão
ONNX_DATA_TYPE_UINT16	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Autoconversão de valores output

Tipo ONNX (tipo de elemento tensor)	Tipo MQL5, suportado pela autoconversão
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex Se um elemento do tensor for zero, o elemento Dados será definido como 0; caso contrário, será definido como 1.
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	Veja. ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

© 2000-2024, MetaQuotes Ltd.

Veja também

Coerção de tipos

© 2000-2024, MetaQuotes Ltd.

Criação de modelo

Existem várias maneiras de obter um modelo ONNX pronto para uso. Uma biblioteca reconhecida é a ONNX Model Zoo, que contém vários modelos ONNX pré-treinados para diferentes tipos de tarefas. A vantagem dessa coleção é que o notebook de cada modelo contém referências ao conjunto de dados de treinamento, bem como links para o documento científico original que descreve a arquitetura do modelo.

A maioria das plataformas de aprendizado de máquina usa Python. Para instalar o ambiente de execução do ONNX para Python, use um dos seguintes comandos:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Para chamar o ambiente de execução do ONNX em Python, use o seguinte comando

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

Os dados de entrada e saída para usar o modelo devem ser vistos na documentação do modelo correspondente. Ferramentas de visualização como Netron ou WinML Dashboard também podem ser usadas para visualizar o modelo. O ambiente de execução do ONNX também permite consultar os metadados do modelo e seus dados de entrada e saída:.

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

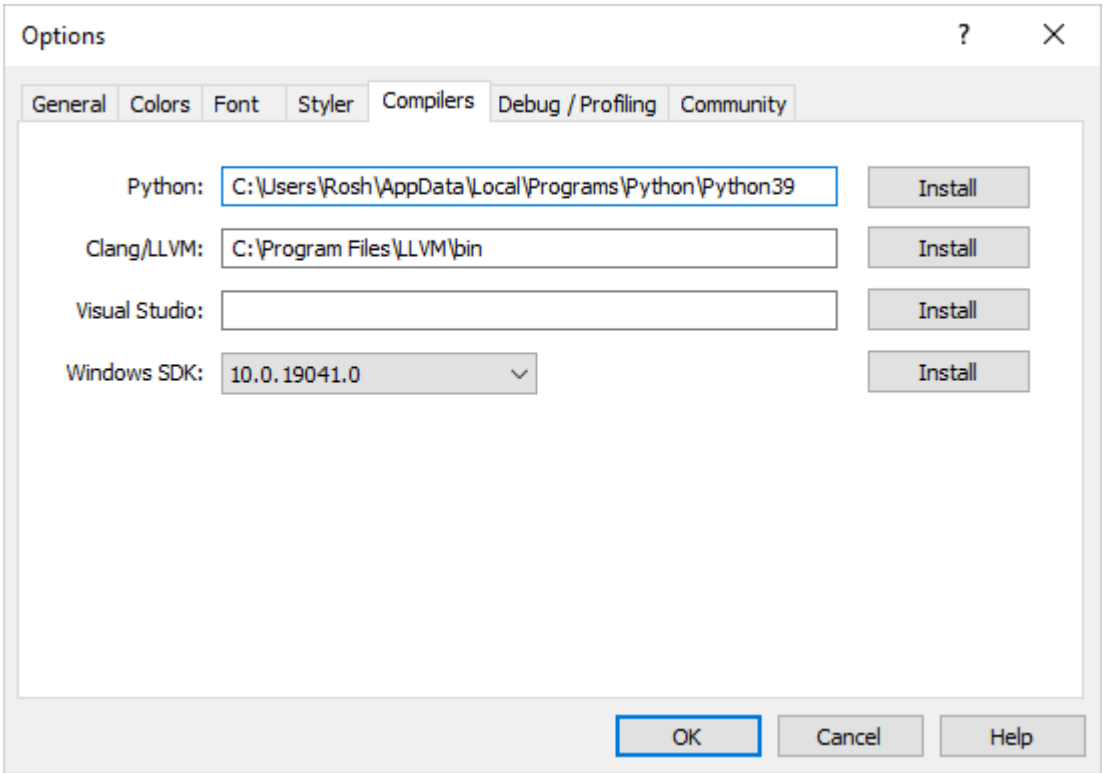
Você pode criar modelos ONNX diretamente no terminal MetaTrader 5 ou no ambiente de desenvolvimento MetaEditor em Python.

Python no MetaTrader 5

A plataforma MetaTrader 5 suporta o trabalho com scripts Python assim que retirados da embalagem. Para isso, os desenvolvedores do terminal disponibilizam o módulo MetaTrader 5 para Python - https://pypi.org/project/MetaTrader5.

O ambiente de desenvolvimento integrado MetaEditor permite não apenas escrever aplicativos em MQL5, mas também executar scripts Python diretamente do editor. Para fazer isso, especifique o caminho para o arquivo executável nas configurações do MetaEditor:.

© 2000-2024, MetaQuotes Ltd.



Se você não tiver o Python em seu computador, clique em "Instalar" para baixar o arquivo de instalação.

Você pode criar um script Python no MetaEdtior ou carregá-lo na pasta de dados do terminal e executá-lo imediatamente pressionando F7 ("Compilar"). Depois disso, o terminal MetaTrader 5 será aberto e o script será rodado no gráfico atual. As mensagens do console Python (stdout, stderr) serão exibidas na seção "Error".

Como trabalhar com modelos no MetaTrader 5

A linguagem MQL5 permite que você execute modelos ONNX diretamente no terminal MetaTrader 5. Isso pode ser feito em três etapas:

1. Treinar um modelo em uma plataforma de terceiros, por exemplo, em Python.
2. Conversão do modelo no ONNX
3. Incorpore o modelo ONNX no EA usando funções ONNX e execute no terminal do MetaTrader 5.

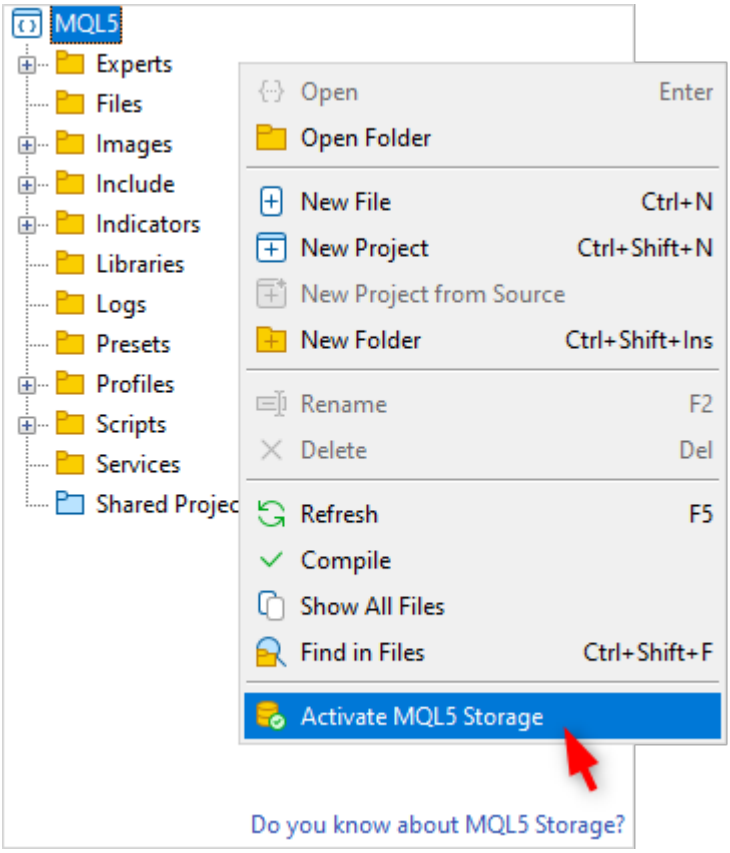
Integração Python da linguagem MQL5 permite executar um script python e salvar o modelo ONNX no MetaEditor ou até mesmo executá-lo em um gráfico no MetaTrader 5. Você pode treinar seu modelo usando um script escrito em Python com a frequência desejada diretamente no terminal. Como a biblioteca contém funções prontas incorporadas para obter dados de preços com os quais você pode alimentar o modelo ONNX:

- copy_rates_from - recebe barras a partir de uma determinada data;
- copy_rates_from_pos - recebe barras a partir de um determinado índice;
- copy_rates_range - recebe barras no intervalo de datas especificado;
- copy_ticks_from - recebe ticks a partir de uma determinada data;
- copy_ticks_range - recebe ticks referentes ao intervalo de datas especificado.

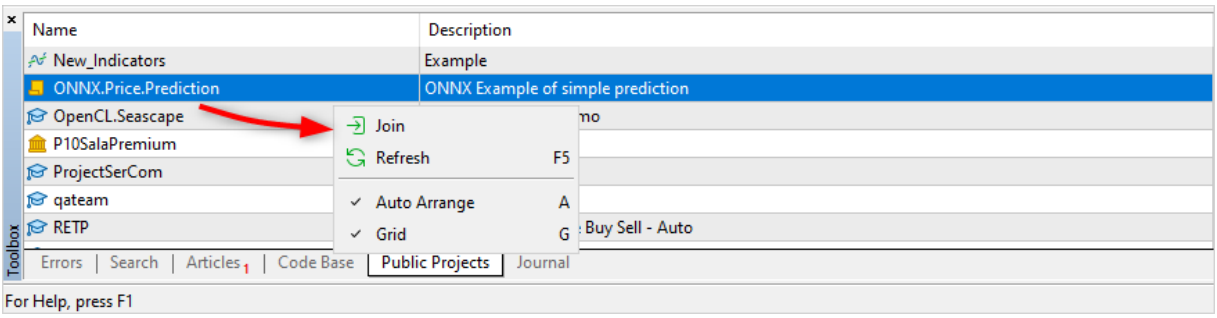
© 2000-2024, MetaQuotes Ltd.

Exemplo de um modelo pronto para uso

Você pode encontrar esse exemplo de modelo ONNX pronto em projetos públicos. Para fazer isso, primeiro ative o MQL5 Storage no Navegador, especificando seu login MQL5 nas configurações do MetaEditor, diferenciando maiúsculas de minúsculas.

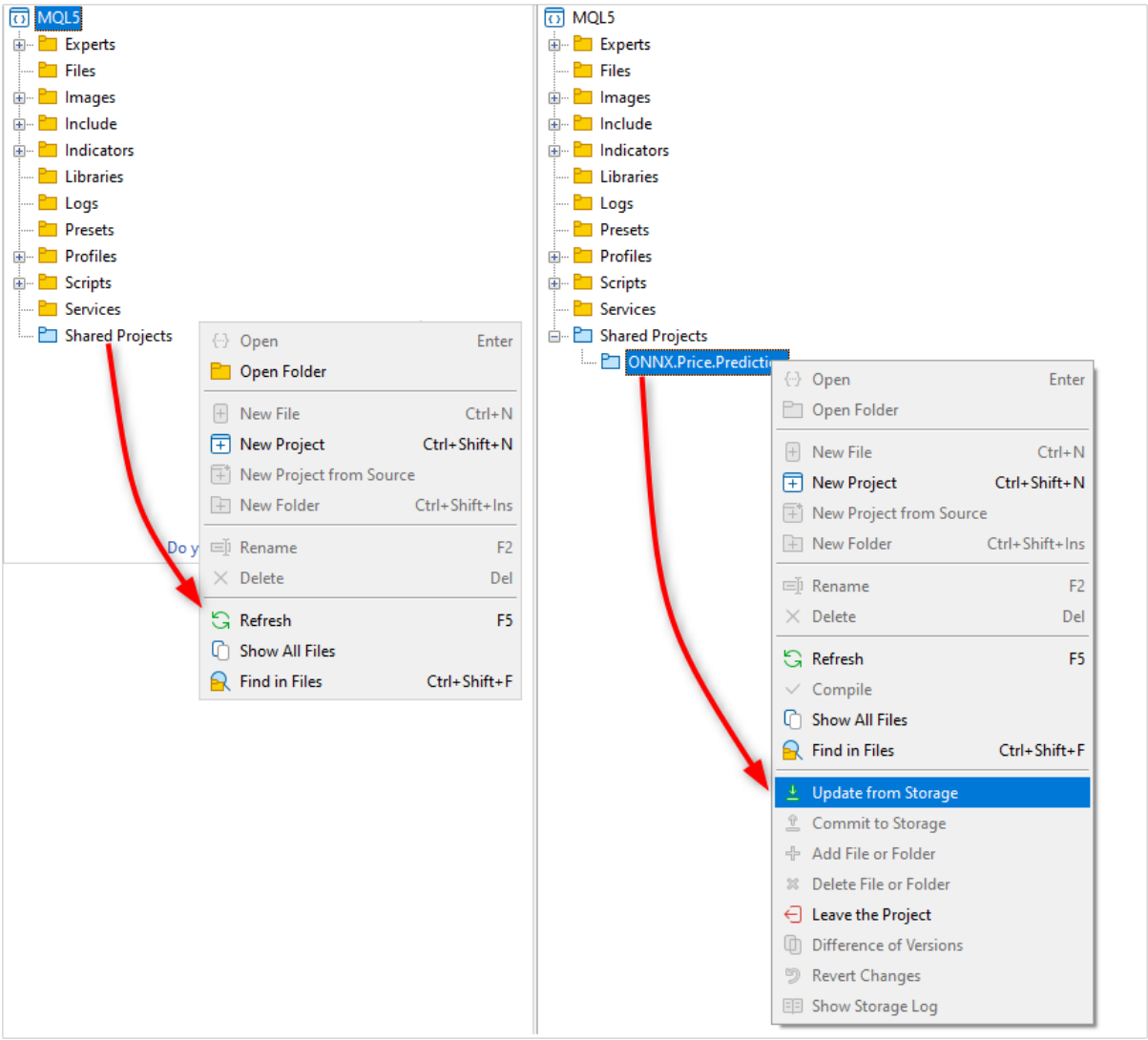


Após a ativação, localize o projeto ONNX.Price.Prediction e acesse-o por meio do comando do menu de contexto.



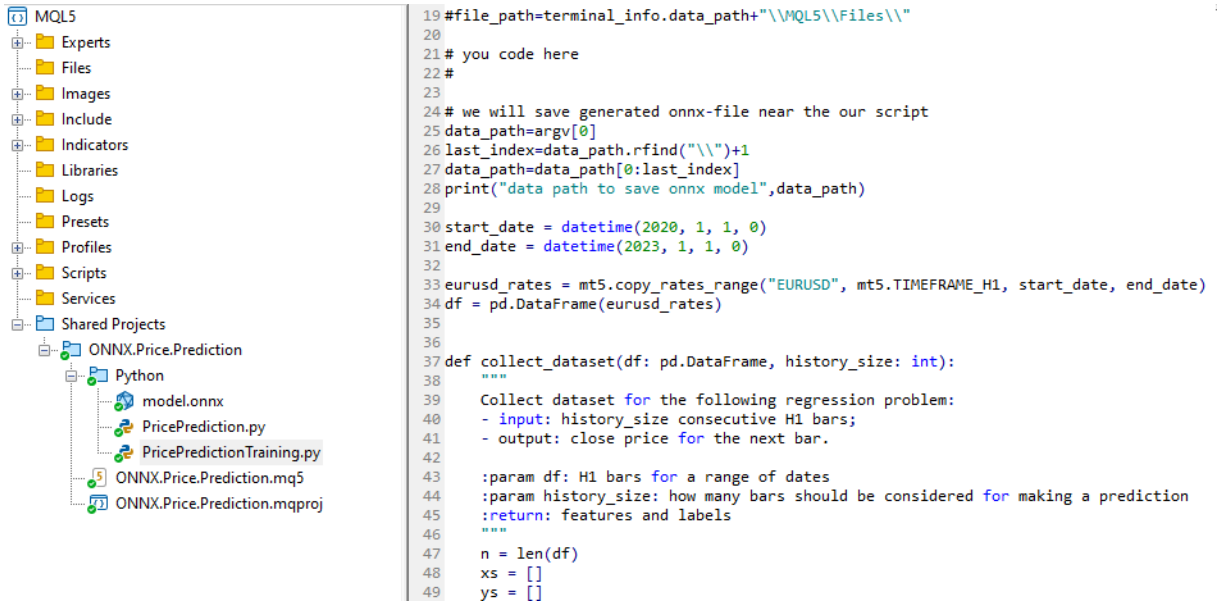
Em seguida, atualize o projeto a partir do MQL5 Storage.

© 2000-2024, MetaQuotes Ltd.



Depois disso, o projeto estará disponível, você encontrará nele um modelo ONNX pronto, dois scripts python, um script MQL5 para rodar o projeto e o arquivo de projeto MQL5 (ONNX.Price.Prediction.mqproj).

© 2000-2024, MetaQuotes Ltd.

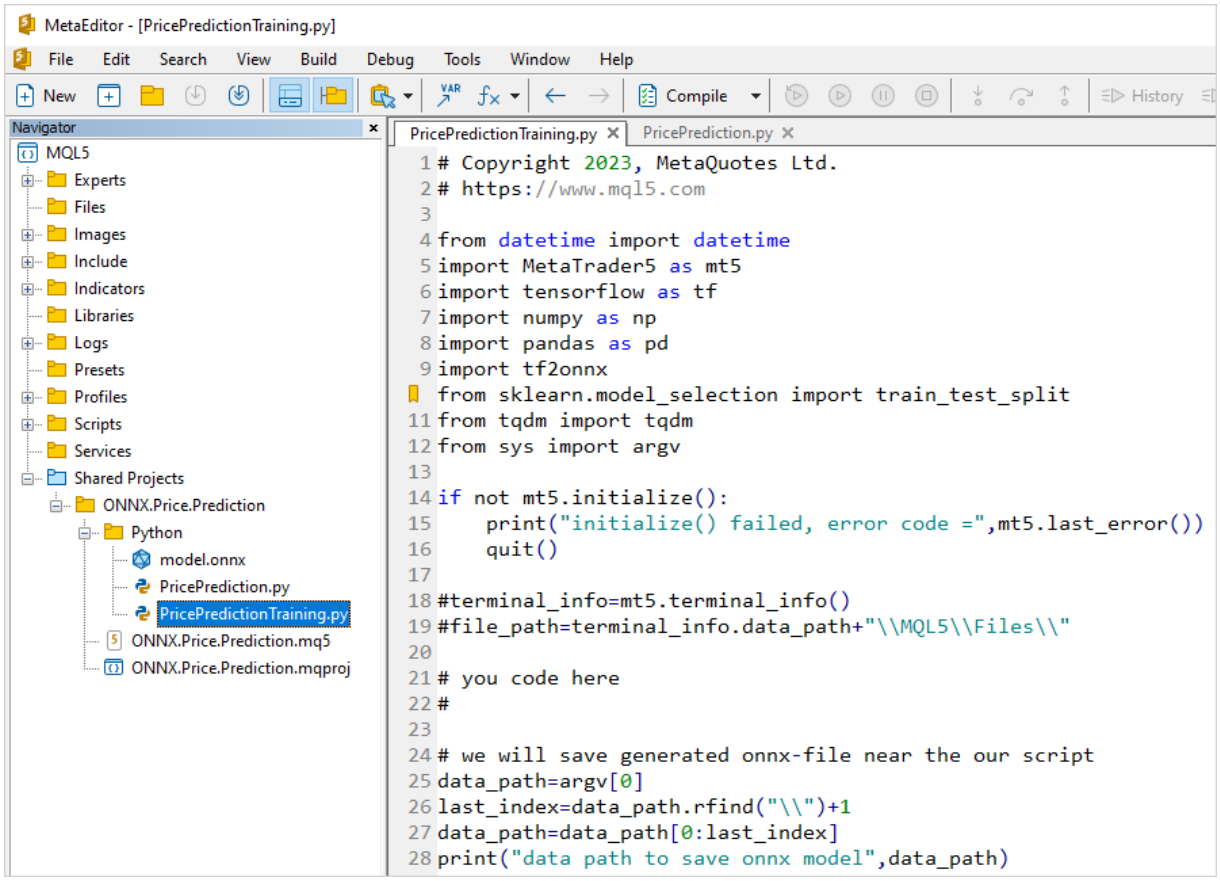


Você mesmo pode criar um modelo ONNX usando o script PricePredictionTraining.py incluído no projeto. Para fazer isso, primeiro instale os módulos necessários a partir da linha de comando em seu computador.

```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

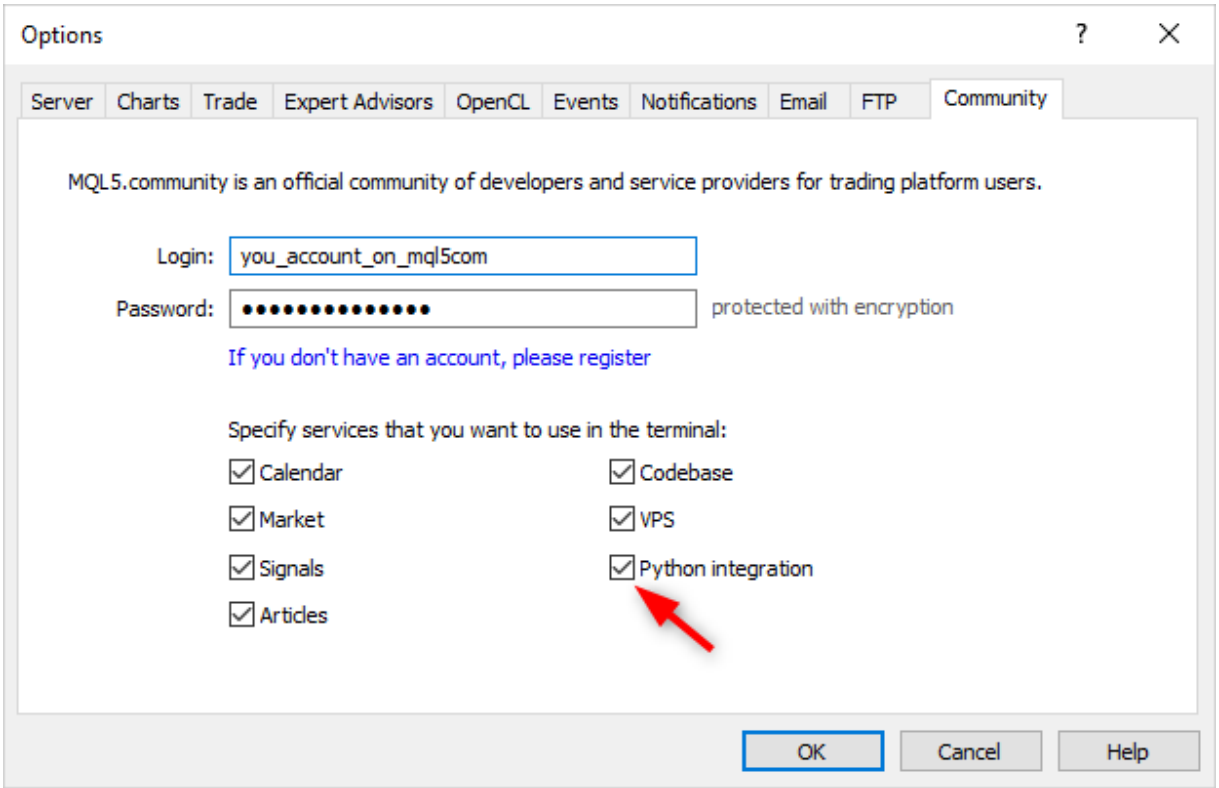
Depois de instalar os módulos, abra o script PricePredictionTraining.py no MetaEditor e execute-o pressionando o botão "Compile" ou a tecla F7.

© 2000-2024, MetaQuotes Ltd.

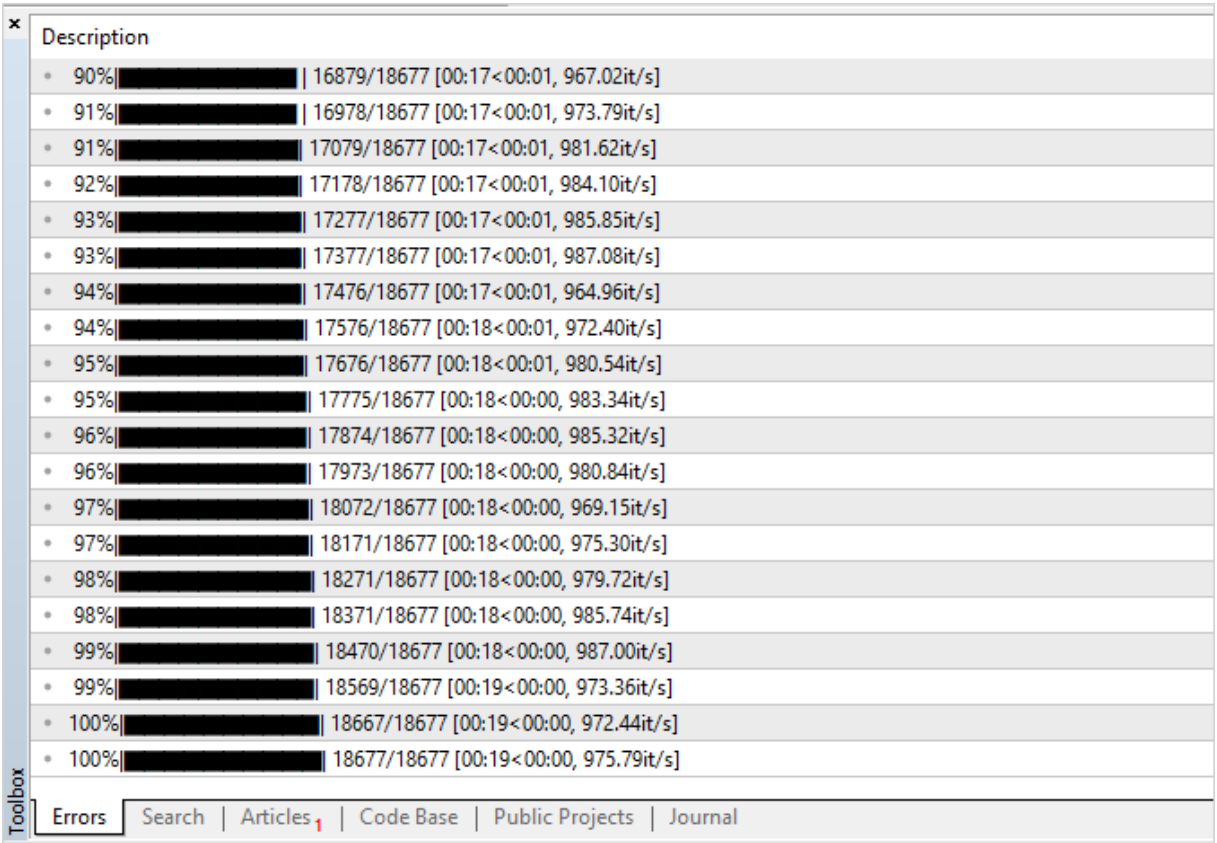


Antes de executar o script python, confira se seu terminal MetaTrader 5 está conectado ao servidor com o símbolo EURUSD. Por exemplo, conecte-se ao servidor MetaQuotes-Demo e verifique a caixa de seleção "Integração com Python" nas configurações do terminal.

© 2000-2024, MetaQuotes Ltd.



Durante o treinamento da rede, o MetaEditor exibirá mensagens do script Python até que o treinamento seja concluído.



© 2000-2024, MetaQuotes Ltd.

Quando o resultado for 100%, o modelo ONNX estará pronto e será salvo na pasta do projeto <pasta de dados do terminal>\MQL5\Shared Projects\ONNX.Price.Prediction\Python.

Verifique o modelo resultante executando o segundo script PricePrediction.py clicando em **F7**.

Time	Source	Message
2023.03.09 15:11:32.149	Python	[[[1.055292 1.05606 1.054948 1.0556]]]
2023.03.09 15:11:32.149	Python	[[[0.0006845 0.00097058 0.00066865 0.00074513]]]
2023.03.09 15:11:32.149	Python	[[[-1.79986207 -1.24668091 -1.50750979 -1.42256891]
2023.03.09 15:11:32.149	Python	[-1.09861711 -0.99940536 -0.90929162 -0.91259138]
2023.03.09 15:11:32.149	Python	[-0.52885558 -0.74182666 -0.53540526 -0.55023892]
2023.03.09 15:11:32.149	Python	[-0.14901455 -0.52546055 0.07776836 -0.48313661]
2023.03.09 15:11:32.149	Python	[-0.0759682 -0.58727944 -0.16151891 -0.63076169]
2023.03.09 15:11:32.149	Python	[-0.29510726 -0.10303148 0.00299109 0.04026138]
2023.03.09 15:11:32.149	Python	[0.5084026 0.278185 -0.19142981 0.20130692]
2023.03.09 15:11:32.149	Python	[0.66910457 0.94788962 0.15254563 0.33551154]
2023.03.09 15:11:32.149	Python	[0.82980654 0.83455499 0.58625381 1.39572799]
2023.03.09 15:11:32.149	Python	[1.94011106 2.14305478 2.48559649 2.02648968]]]
2023.03.09 15:11:32.181	Python	[[1.9743274]]
2023.03.09 15:11:32.181	Python	predict: [1.05707]

Toolbox | Errors | Search | Articles 1 | Code Base | Public Projects | Journal

For Help, press F1

Inicialização do modelo

Para utilizar o modelo ONNX em MQL5, é necessário seguir três etapas:

1. Carregá-lo a partir do arquivo *.onnx usando a função OnnxCreate ou a partir de um array usando OnnxCreateFromBuffer.
2. Especificar a forma dos dados de entrada e saída por meio das funções OnnxSetInputShape e OnnxSetOutputShape.
3. Iniciar o modelo usando OnnxRun, após lhe dar parâmetros de entrada e saída.
4. Encerrar o modelo, se necessário, com OnnxRelease.

Ao criar o modelo ONNX, é necessário considerar os limites e restrições existentes descritos em https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md

Por exemplo, aqui estão algumas delas:

Operação	Tipos de dados suportados
ReduceSum	tensor(double), tensor(float), tensor(int32), tensor(int64)
Mul	tensor(bfloat16), tensor(double), tensor(float), tensor(float16), tensor(int32), tensor(int64), tensor(uint32), tensor(uint64)

Veja abaixo um exemplo de código MQL5 extraído do projeto público ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};    // forma dos dados de saída do modelo
const long   ExtInputShape [] = {1,10,4}; // forma dos dados de entrada do modelo
#resource "Python/model.onnx" as uchar ExtModel[]// modelo em forma de recurso
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- obtemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- alimentamos com um conjunto de vetores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- preenchemos matrizes de normalização
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- normalizamos os dados de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- criamos o modelo
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- especificamos a forma dos dados de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- especificamos a forma dos dados de saída
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertemos dados de entrada normalizados em dados do tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- obtemos o resultado do modelo - a previsão de preço
   vectorf y_norm(1);
//--- iniciamos o modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- registramos no log o valor de saída do modelo
   Print(y_norm);
//--- realizamos uma transformação inversa para obter o preço previsto
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- concluímos o trabalho
   OnnxRelease(handle);
   return(0);
  }
```

Veja um exemplo de execução do script:

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
```

© 2000-2024, MetaQuotes Ltd.

```
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

O terminal MetaTrader 5 seleciona de forma independente o melhor executor para os cálculos — ONNX Runtime Execution Provider. Neste caso, o modelo realiza seu trabalho usando a CPU.

Vamos modificar o script para calcular a porcentagem de previsões bem-sucedidas do preço de fechamento com base nos valores das 10 barras anteriores.

```
#resource "Python/model.onnx" as uchar ExtModel[]// modelo em forma de recurso

#define TESTS 10000  // número de amostras de teste
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- criamos o modelo
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- como o modelo não tem dimensões de tensor de entrada definidas, devemos defini-l
//--- o primeiro índice é o tamanho do pacote; o segundo índice, tamanho da série; o t
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- como o modelo não tem dimensões de tensor de saída definidas, devemos defini-las
//--- o primeiro índice é o tamanho do lote, que deve corresponder ao tamanho do lote
//--- o segundo índice é o número de preços previstos (prevemos apenas o Close)
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
//--- iniciamos os testes
```

© 2000-2024, MetaQuotes Ltd.

```
   vector closes(TESTS);       // vetor para armazenar preços de verificação
   vector predicts(TESTS);     // vetor para armazenar as previsões resultantes
   vector prev_closes(TESTS); // vetor para armazenar os penúltimos preços

   matrix rates;               // matriz para obter uma série OHLC
   matrix splitted[2];         // duas submatrizes para dividir a série em uma de teste
   ulong  parts[]={10,1};      // dimensões dos subarrays separáveis

//--- começamos a partir da barra anterior
   for(int i=1; i<=TESTS; i++)
     {
      //--- obtemos 11 barras
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- dividimos a matriz em uma de teste e outra de verificação
      rates.Vsplit(parts,splitted);
      //--- pegamos o preço Close a partir da matriz de verificação
      closes[i-1]=splitted[1][3][0];
      //--- último Close na série de teste
      prev_closes[i-1]=splitted[0][3][9];

      //--- enviamos ao teste uma matriz de teste de 10 barras
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- erro de execução
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- concluímos o trabalho
   OnnxRelease(session_handle);
//--- avaliamos se o movimento de preço previsto está correto
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 && delta_
         right_directions++;
   PrintFormat("right direction predictions = %.2f%%",(right_directions*100.0)/double
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  Preparando os dados e utilizando o modelo                       |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
   static matrixf input_data(10,4); // matriz para dados de entrada transformados
```

```
   static vectorf output_data(1);   // vetor para obter o resultado
   static matrix mm(10,4);          // matriz de vetores horizontais Mean
   static matrix ms(10,4);          // matriz de vetores horizontais Std

//--- o modelo deve ser alimentado com um conjunto de vetores OHLC verticais
   matrix x_norm=rates.Transpose();
//--- normalizamos os preços
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//--- iniciamos o modelo
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- revertemos a normalização do preço a partir do valor de saída
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Executamos o script e obtemos uma precisão de previsão de cerca de 51%

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
right direction predictions = 51.34 %
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Verificação do modelo no testador de estratégia

Os modelos para mercados financeiros podem ser testados no testador de estratégias do terminal MetaTrader 5. Essa é a opção mais rápida e conveniente que não exige esforços adicionais para emular o ambiente de mercado e as condições de negociação.

Reescreveremos o código do projeto público ONNX.Price.Prediction no EA para verificar o modelo. Isso requer pequenas edições.

Transferimos a criação do modelo para a função OnInit e encerramos a sessão onnx em OnDeinit. Colocamos o bloco principal de manipulação do modelo no manipulador OnTick.

Também adicionamos uma função para obter o preço de fechamento das duas barras anteriores a fim de comparar o preço de fechamento real e a previsão.

O código do EA é pequeno e fácil de ler.

```
const long   ExtInputShape [] = {1,10,4}; // forma dos dados de entrada do modelo
const long   ExtOutputShape[] = {1,1};    // forma dos dados de saída do modelo
#resource "Python/model.onnx" as uchar ExtModel[];// modelo em forma de recurso

long handle;         // identificador do modelo
ulong predictions=0; // contador de previsões
ulong confirmed=0;   // contador de previsões bem-sucedidas
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- verificações básicas
   if(_Symbol!="EURUSD")
     {
      Print("Symbol must be EURUSD, testing aborted");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("Timeframe must be H1, testing aborted");
      return(-1);
     }
//--- criamos o modelo
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- especificamos a forma dos dados de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

© 2000-2024, MetaQuotes Ltd.

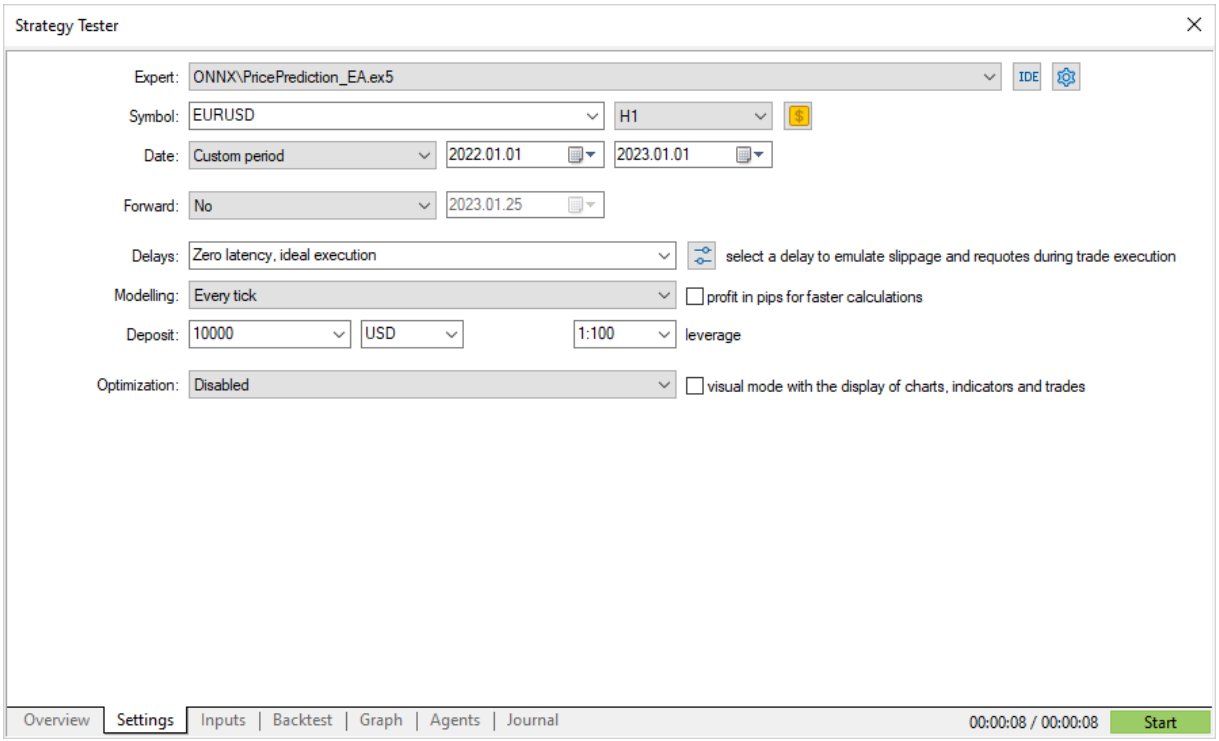
```
//--- especificamos a forma dos dados de saída
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- concluímos o trabalho do modelo
   OnnxRelease(handle);
//--- calculamos e exibimos estatísticas de previsão
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions))
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- verificamos o tempo de abertura da barra atual
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- se o tempo de abertura da barra não muda, então saímos antes da próxima chamada
   if(time==open_time)
      return;
//--- obtemos os preços de fechamento das últimas 2 barras concluídas
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("CopyClose(2 bars) failed, error %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // movimento de preço previsto
   double delta_actual=close[1]-close[0]; // variação real de preços
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

© 2000-2024, MetaQuotes Ltd.

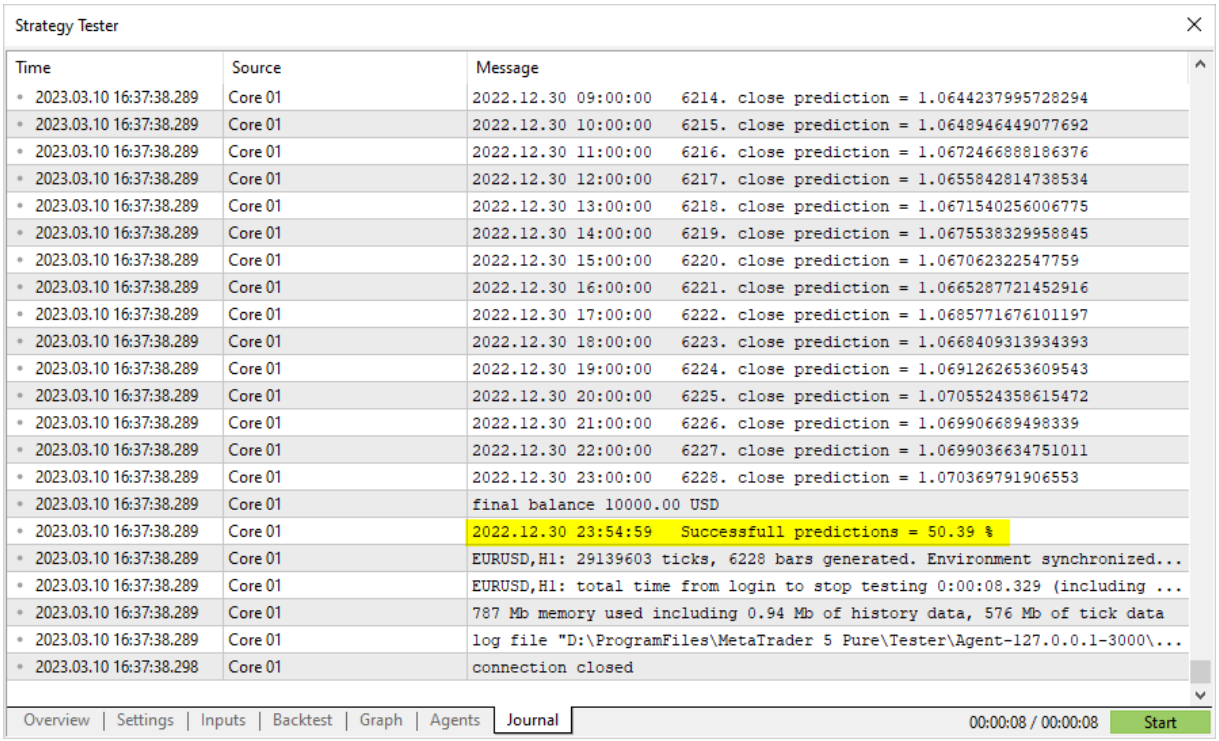
```
//--- calculamos o preço de fechamento em uma nova barra para verificar na próxima ba
   matrix rates;
//--- obtemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- alimentamos com um conjunto de vetores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- preenchemos matrizes de normalização
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalizamos os dados de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- convertemos dados de entrada normalizados em dados do tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- obtemos o resultado do modelo - a previsão de preço
   vectorf y_norm(1);
//--- iniciamos o modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- realizamos uma transformação inversa para obter o preço previsto e verificamos e
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // aumentamos o contador de previsões
   Print(predictions,". close prediction = ",predict);
//--- memorizamos o tempo de abertura da barra para verificar no próximo tick
   open_time=time;
  }
```

Compilamos o Expert Advisor e começamos a testar no intervalo 2022, especificando o símbolo EURUSD e o período de tempo H1 com os quais o modelo foi treinado. O modo de simulação de ticks não importa, pois o código tem uma verificação do aparecimento de nova barra.

© 2000-2024, MetaQuotes Ltd.



Executamos e obtemos o resultado no log de teste — pouco mais de 50% das previsões corretas para 2022.



Strategy Tester

Time	Source	Message
2023.03.10 16:37:38.289	Core 01	2022.12.30 09:00:00 6214. close prediction = 1.0644237995728294
2023.03.10 16:37:38.289	Core 01	2022.12.30 10:00:00 6215. close prediction = 1.0648946449077692
2023.03.10 16:37:38.289	Core 01	2022.12.30 11:00:00 6216. close prediction = 1.0672466888186376
2023.03.10 16:37:38.289	Core 01	2022.12.30 12:00:00 6217. close prediction = 1.0655842814738534
2023.03.10 16:37:38.289	Core 01	2022.12.30 13:00:00 6218. close prediction = 1.0671540256006775
2023.03.10 16:37:38.289	Core 01	2022.12.30 14:00:00 6219. close prediction = 1.0675538329958845
2023.03.10 16:37:38.289	Core 01	2022.12.30 15:00:00 6220. close prediction = 1.067062322547759
2023.03.10 16:37:38.289	Core 01	2022.12.30 16:00:00 6221. close prediction = 1.0665287721452916
2023.03.10 16:37:38.289	Core 01	2022.12.30 17:00:00 6222. close prediction = 1.0685771676101197
2023.03.10 16:37:38.289	Core 01	2022.12.30 18:00:00 6223. close prediction = 1.0668409313934393
2023.03.10 16:37:38.289	Core 01	2022.12.30 19:00:00 6224. close prediction = 1.0691262653609543
2023.03.10 16:37:38.289	Core 01	2022.12.30 20:00:00 6225. close prediction = 1.0705524358615472
2023.03.10 16:37:38.289	Core 01	2022.12.30 21:00:00 6226. close prediction = 1.069906689498339
2023.03.10 16:37:38.289	Core 01	2022.12.30 22:00:00 6227. close prediction = 1.0699036634751011
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:00:00 6228. close prediction = 1.070369791906553
2023.03.10 16:37:38.289	Core 01	final balance 10000.00 USD
2023.03.10 16:37:38.289	Core 01	2022.12.30 23:54:59 Successfull predictions = 50.39 %
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: 29139603 ticks, 6228 bars generated. Environment synchronized...
2023.03.10 16:37:38.289	Core 01	EURUSD,H1: total time from login to stop testing 0:00:08.329 (including ...
2023.03.10 16:37:38.289	Core 01	787 Mb memory used including 0.94 Mb of history data, 576 Mb of tick data
2023.03.10 16:37:38.289	Core 01	log file "D:\ProgramFiles\MetaTrader 5 Pure\Tester\Agent-127.0.0.1-3000\...
2023.03.10 16:37:38.298	Core 01	connection closed

Overview | Settings | Inputs | Backtest | Graph | Agents | Journal 00:00:08 / 00:00:08 Start

Se os resultados da verificação preliminar do modelo forem satisfatórios, podemos começar a escrever uma estratégia de negociação completa e usá-la.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

OnnxCreate

Cria uma sessão ONNX com carregamento de modelo a partir de um arquivo *.onnx

```
long  OnnxCreate(
   string  filename,  // caminho do arquivo
   uint    flags      // sinalizadores para criação de um modelo
   );
```

Parâmetros

filename

[in] Caminho para o arquivo *.onnx do modelo em relação à pasta \MQL5\Files\.

flags

[in] Sinalizador de ENUM_ONNX_FLAGS, que descreve o modo de criação do modelo — ONNX_COMMON_FOLDER e ONNX_DEBUG_LOGS.

Valor retornado

Identificador da sessão criada ou INVALID_HANDLE em caso de erro. Para obter o código do erro chame a função GetLastError.

Observação

Se o arquivo especificado não for encontrado no disco, será feita uma segunda tentativa de abrir o arquivo com a extensão '.onnx' anexada ao nome.

© 2000-2024, MetaQuotes Ltd.

OnnxCreateFromBuffer

Cria uma sessão ONNX com carregamento de modelo a partir de uma matriz de dados

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],    // referência para uma matriz
   ulong         flags        // sinalizadores para criação de um modelo
   );
```

Parâmetros

buffer

[in] Matriz com dados do modelo ONNX.

flags

[in] Sinalizador de ENUM_ONNX_FLAGS, que descreve o modo de criação do modelo — ONNX_COMMON_FOLDER e ONNX_DEBUG_LOGS.

Valor retornado

Identificador da sessão criada ou INVALID_HANDLE em caso de erro. Para obter o código do erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxRelease

Exclusão de sessão ONNX.

```
bool  OnnxRelease(
   long   onnx_handle  // Identificador de sessão ONNX
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxRun

Execução do modelo ONNX.

```
bool  OnnxRun(
   long    onnx_handle,  // identificador de sessão
   ulong   flags,        // sinalizadores que descrevem o modo de execução
   ...                   // parâmetros de entrada e saída do modelo
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

flags

[in] Sinalizador de ENUM_ONNX_FLAGS, que descreve o modo de execução — ONNX_DEBUG_LOGS e ONNX_NO_CONVERSION.

...

[in] [out] Parâmetros de entrada e saída do modelo.

Retorna true se for bem-sucedido, caso contrário, false. Para obter o código de erro chame a função GetLastError.

ENUM_ONNX_FLAGS

Identificador	Descrição
ONNX_DEBUG_LOGS	Saída de registros de depuração
ONNX_NO_CONVERSION	Proibir a autoconversão, usar os dados do usuário como estão
ONNX_COMMON_FOLDER	Carga do arquivo do modelo a partir da pasta Common/Files, igual em valor ao sinalizador FILE_COMMON

Exemplo:

```
const long                              ExtOutputShape[] = {1,1};     // forma dos dados
const long                              ExtInputShape [] = {1,10,4}; // forma dos dados
#resource "Python/model.onnx" as uchar ExtModel[]                    // modelo em forma
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- obtemos 10 barras
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- alimentamos com um conjunto de vetores OHLC
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- preenchemos matrizes de normalização
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- normalizamos os dados de entrada
   x_norm-=mm;
   x_norm/=ms;
//--- criamos o modelo
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- especificamos a forma dos dados de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- especificamos a forma dos dados de saída
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- convertemos dados de entrada normalizados em dados do tipo float
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- obtemos o resultado do modelo - a previsão de preço
   vectorf y_norm(1);
//--- iniciamos o modelo
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- registramos no log o valor de saída do modelo
   Print(y_norm);
//--- realizamos uma transformação inversa para obter o preço previsto
```

© 2000-2024, MetaQuotes Ltd.

```
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- concluímos o trabalho
   OnnxRelease(handle);
   return(0);
  };
```

Veja também

OnnxSetInputShape, OnnxSetOutputShape

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputCount

Obtenção do número de parâmetros de entrada do modelo ONNX.

```
long  OnnxGetInputCount(
   long   onnx_handle  // Identificador de sessão ONNX
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

Valor retornado

Se for bem-sucedido, retorna o número de parâmetros de entrada; caso contrário, retorna -1. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputCount

Obtenção do número de parâmetros de saída do modelo ONNX.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // Identificador de sessão ONNX
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

Valor retornado

Se for bem-sucedido, retorna o número de parâmetros de saída; caso contrário, retorna -1. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputName

Obtenção do nome do parâmetro de entrada do modelo por índice.

```
string  OnnxGetInputName(
   long   onnx_handle,  // identificador de sessão
   long   index         // índice de parâmetro
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Índice do parâmetro de entrada, a partir de 0.

Valor retornado

Retorna o nome do parâmetro de entrada se for bem-sucedido; caso contrário, retorna NULL. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputName

Obtenção do nome do parâmetro de saída do modelo por índice.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // identificador de sessão
   long   index         // índice de parâmetro
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Índice do parâmetro de saída, a partir de 0.

Valor retornado

Retorna o nome do parâmetro de saída se for bem-sucedido; caso contrário, retorna NULL. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetInputTypeInfo

Obtenção de uma descrição do tipo de parâmetro de entrada do modelo.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // identificador de sessão
   long           index,        // índice de parâmetro
   OnnxTypeInfo&  typeinfo      // descrição do tipo de parâmetro
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Índice do parâmetro de entrada, a partir de 0.

typeinfo

[out] Estrutura OnnxTypeInfo, que descreve o tipo de parâmetro de entrada.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

Obtenção de uma descrição do tipo de parâmetro de saída do modelo.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // identificador de sessão
   long           index,        // índice de parâmetro
   OnnxTypeInfo&  typeinfo      // descrição do tipo de parâmetro
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

index

[in] Índice do parâmetro de saída, a partir de 0.

typeinfo

[out] Estrutura OnnxTypeInfo, que descreve o tipo de parâmetro de saída.

Valor retornado

Retorna true se for bem-sucedido, caso contrário, false. Para obter o código de erro chame a função GetLastError.

© 2000-2024, MetaQuotes Ltd.

OnnxSetInputShape

Define a dimensionalidade dos dados de entrada do modelo por índice.

```
bool  OnnxSetInputShape(
   long          onnx_handle,  // identificador de sessão
   long          input_index,  // índice do parâmetro de entrada
   const ulong&  shape[]       // matriz que descreve a dimensionalidade dos dados de
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

input_index

[in] Índice do parâmetro de entrada, a partir de 0.

shape

[in] Matriz que descreve a dimensionalidade dos dados de entrada.

Valor retornado

Retorna o nome do parâmetro de entrada se for bem-sucedido; caso contrário, retorna NULL. Para obter o código de erro chame a função GetLastError.

Exemplo:

```
//---- descrevemos a dimensionalidade dos dados de entrada do modelo
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- criamos o modelo
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- especificamos a forma dos dados de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- especificamos a forma dos dados de saída
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Veja também

© 2000-2024, MetaQuotes Ltd.

OnnxSetOutputShape

© 2000-2024, MetaQuotes Ltd.

OnnxSetOutputShape

Define a dimensionalidade dos dados de saída do modelo por índice.

```
bool  OnnxSetOutputShape(
   long          onnx_handle,    // identificador de sessão
   long          output_index,   // índice do parâmetro de saída
   const ulong&  shape[]         // matriz que descreve a dimensionalidade dos dados de
   );
```

Parâmetros

onnx_handle

[in] Identificador do objeto da sessão ONNX, criado por meio de OnnxCreate ou OnnxCreateFromBuffer.

output_index

[in] Índice do parâmetro de saída, a partir de 0.

shape

[in] Matriz que descreve a dimensionalidade dos dados de saída.

Valor retornado

Retorna o nome do parâmetro de entrada se for bem-sucedido; caso contrário, retorna NULL. Para obter o código de erro chame a função GetLastError.

Exemplo:

```
//---- descrevemos a dimensionalidade dos dados de entrada do modelo
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- criamos o modelo
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- especificamos a forma dos dados de entrada
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- especificamos a forma dos dados de saída
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Veja também

© 2000-2024, MetaQuotes Ltd.

OnnxSetInputShape

© 2000-2024, MetaQuotes Ltd.

Estruturas de dados

As seguintes estruturas de dados são usadas para trabalhar com modelos ONNX:

OnnxTypeInfo

A estrutura descreve o tipo de parâmetro de entrada ou saída do modelo ONNX

```
struct OnnxTypeInfo
  {
   ENUM_ONNX_TYPE          type;           // tipo de parâmetro
   OnnxTensorTypeInfo      tensor;         // descrição do tensor
   OnnxMapTypeInfo         map;            // descrição do mapa
   OnnxSequenceTypeInfo    sequence;       // descrição da sequência
  };
```

Somente o tensor (ONNX_TYPE_TENSOR) pode ser usado como parâmetro de entrada; nesse caso, somente o campo OnnxTypeInfo::tensor é preenchido com valores; os outros campos (mapa e sequência) permanecem indefinidos.

Apenas um dos três tipos OnnxTypeInfo (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP ou ONNX_TYPE_SEQUENCE) pode ser usado como parâmetro de saída, dependendo do tipo, será preenchida a subestrutura apropriada (OnnxTypeInfo::tensor, OnnxTypeInfo::map ou OnnxTypeInfo::sequence).

OnnxTensorTypeInfo

A estrutura descreve o tensor no parâmetro de entrada ou saída do modelo ONNX

```
struct OnnxTensorTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    data_type;      // tipo de dados no tensor
   const long                   dimensions[];   // número de elementos no tensor
  };
```

OnnxMapTypeInfo

A estrutura descreve o mapa que é obtido no parâmetro de saída do modelo ONNX

```
struct OnnxMapTypeInfo
  {
   const ENUM_ONNX_DATA_TYPE    key_type;       // tipo de chave
   const OnnxTypeInfo&          value_type;     // tipo de valor
  };
```

OnnxSequenceTypeInfo

A estrutura descreve a sequência que resulta no parâmetro de saída do modelo ONNX

© 2000-2024, MetaQuotes Ltd.

```
struct OnnxSequenceTypeInfo
  {
   const OnnxTypeInfo&          value_type;     // tipo de dados da sequência
  };
```

ENUM_ONNX_TYPE

A enumeração **ENUM_ONNX_TYPE** descreve o tipo de parâmetro do modelo

Identificador	Descrição
ONNX_TYPE_UNKNOWN	Desconhecido
ONNX_TYPE_TENSOR	Tensor
ONNX_TYPE_SEQUENCE	Sequência
ONNX_TYPE_MAP	Mapa
ONNX_TYPE_OPAQUE	Abstrato
ONNX_TYPE_SPARSETENSOR	Tensor esparso

ENUM_ONNX_DATA_TYPE

A enumeração **ENUM_ONNX_DATA_TYPE** descreve o tipo de dados usados

Identificador	Descrição
ONNX_DATA_TYPE_UNDEFINED	Não definido
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	int de 8 bits
ONNX_DATA_TYPE_UINT16	uint de 16 bits
ONNX_DATA_TYPE_INT16	int de 16 bits
ONNX_DATA_TYPE_INT32	int de 32 bits
ONNX_DATA_TYPE_INT64	int de 64 bits
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	float de 16 bits
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint de 32 bits
ONNX_DATA_TYPE_UINT64	uint de 64 bits
ONNX_DATA_TYPE_COMPLEX64	Número complexo de 64 bits

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
ONNX_DATA_TYPE_COMPLEX128	Número complexo de 128 bits
ONNX_DATA_TYPE_BFLOAT16	bfloat de 16 bits (Brain Floating Point)

ENUM_ONNX_FLAGS

A enumeração **ENUM_ONNX_FLAGS** descreve o modo de inicialização do modelo

Identificador	Descrição
ONNX_DEBUG_LOGS	Saída de registros de depuração
ONNX_NO_CONVERSION	Proibir a autoconversão, usar os dados do usuário como estão
ONNX_COMMON_FOLDER	Carga do arquivo do modelo a partir da pasta Common/Files, igual em valor ao sinalizador FILE_COMMON

© 2000-2024, MetaQuotes Ltd.

Biblioteca Padrão

Este grupo de capítulos contém os detalhes técnicos da Biblioteca Padrão MQL5 e as descrições de todos os seus componentes fundamentais.

A Biblioteca Padrão MQL5 é escrita em linguagem MQL5 e projetada para facilitar a escrita de programas (indicadores, scripts, experts) para os usuários finais. A biblioteca oferece acesso prático para a maioria das funções internas MQL5.

A Biblioteca Padrão MQL5 é alocado no diretório de trabalho do terminal e na pasta "Include".

Seção	Localização
Matemática	Include\Math\
OpenCL	Include\OpenCL\
Classe base CObject	Include\
Coleções de dados	Include\Arrays\
Coleções de dados genéricas	Include\Generic\
Arquivos	Include\Files\
Linhas	Include\Strings\
Objetos gráficos	Include\Objects\
Gráficos personalizados	Include\Canvas\
Gráfico 3D	Include\Canvas\
Gráficos de preços	Include\Charts\
Gráficos científicos	Include\Graphics\
Indicadores	Include\Indicators\
Classes de negociação	Include\Trade\
Módulos de estratégias	Include\Expert\
Painéis e caixas de diálogo	Include\Controls\

© 2000-2024, MetaQuotes Ltd.

Matemática

Para executar cálculos em diferentes áreas da matemática, são oferecidas várias bibliotecas:

- Estatística – funções para trabalhar com diferentes distribuições a partir da teoria da probabilidade
- Logica difusa – biblioteca de lógica fuzzy que implementa um sistema de inferência fuzzy Mamdani e Sugeno
- ALGLIB – análise de dados (clustering, árvores de decisão, regressão linear, redes neurais), solução de equações diferenciais, análise de Fourier, integração numérica, otimização, análise estatística e muito mais.

© 2000-2024, MetaQuotes Ltd.

Estatísticas

A biblioteca estatística destina-se a um cômodo trabalho com as distribuições estatísticas básicas.

Para cada distribuição na biblioteca, são apresentadas 5 características:

1. Cálculo de densidade da distribuição, função do tipo MathProbabilityDensityX()
2. Cálculo de probabilidade, função do tipo MathCumulativeDistributionX()
3. Cálculo de quantis das distribuições, função do tipo MathQuantileX()
4. Geração de números aleatórios com a distribuição definida, função do tipo MathRandomX()
5. Cálculo de momentos teóricos das distribuições, função do tipo MathMomentsX()

Além de calcular os valores para as variáveis aleatórias individuais, na biblioteca, são apresentadas também as sobrecargas das funções, que produzem os cálculos nas matrizes.

- Características estatísticas
- Distribuição normal
- Distribuição log-normal
- Distribuição beta
- Distribuição beta não-central
- Distribuição gama
- Qui-quadrado
- Qui-quadrado não-central
- Distribuição exponencial
- Distribuição F de Fisher-Snedecor
- Distribuição F de Fisher-Snedecor não-central
- Distribuição t de Student
- Distribuição t de Student não-central
- Distribuição logística
- Distribuição de Cauchy
- Distribuição uniforme
- Distribuição de Weibull
- Distribuição binomial
- Distribuição binomial negativa
- Distribuição geométrica
- Distribuição hipergeométrica
- Distribuição de Poisson
- Funções auxiliares

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                   NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- habilitamos as funções para calcular a distribuição normal
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- definimos os parâmetros da distribuição normal
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Distribuição normal com parâmetros mu=%G e sigma=%G, exemplos de cálcu
//--- definimos o intervalo
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- variáveis para calcular a probabilidade
   double cdf1,cdf2,probability;
//--- variáveis para o código de erro
   int error_code1,error_code2;
//--- calculamos os valores de função da distribuição
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- verificamos o código de erros
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- calculamos a probabilidade da variável aleatória no intervalo
      probability=cdf2-cdf1;
      //--- exibimos o resultado
      PrintFormat("1. Calcular no intervalo %.5f<x<%.5f a variável aleatória",x1,x2);
      PrintFormat("  Resposta: Probability = %5.8f",probability);
     }

//--- Encontramos o intervalo de valores da variável aleatória x, correspondente ao 95
   probability=0.95; //  definimos a probabilidade de confiança
//--- definimos a probabilidade nos limites do i intervalo
   double p1=(1.0- probability)*0.5;
   double p2=probability+(1.0- probability)*0.5;
//--- calculamos os limites do intervalo
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- verificamos o código de erros
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- exibimos o resultado
      PrintFormat("2. Para o intervalo de confiança = %.2f encontrar o intervalo da va
      PrintFormat("  Resposta: intervalo  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Calculamos os 4 primeiros momentos calculados e teóricos da distrib
//--- Geramos a matriz de números aleatórios, calculamos os primeiros 4 momentos e com
   int data_count=1000000;  // definimos o número de valores e preparamos a matriz
   double data[];
   ArrayResize(data,data_count);
//--- geramos os valores aleatórios e salvamo-los na matriz
```

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- definimos o índice do valor inicial e número de dados para o cálculo
   int start=0;
   int count=data_count;
//--- calculamos os primeiros 4 momentos dos valores gerados
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- variáveis para os momentos teóricos
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- exibimos os valores dos momentos calculados
   PrintFormat("            Mean           Variance          Skewness           Kurtos
   PrintFormat("Calculated  %.10f   %.10f      %.10f      %.10f",mean,variance,skewnes
//--- calculamos os valores teóricos dos momentos e comparamos com os obtidos
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Theoretical %.10f   %.10f      %.10f       %.10f",normal_mean,norma
      PrintFormat("Difference  %.10f   %.10f      %.10f       %.10f",mean-normal_mean,
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Características estatísticas

Este grupo de funções calcula as características estatísticas dos elementos na matriz:

- média,
- dispersão,
- obliquidade,
- curtose,
- mediana,
- Desvios padrão e
- padrão amostral

Função	Descrição
MathMean	Calcula a média (primeiro momento) dos elementos da matriz
MathVariance	Calcula a dispersão (segundo momento) dos elementos da matriz
MathSkewness	Calcula a obliquidade (terceiro momento) dos elementos da matriz
MathKurtosis	Calcula a curtose (quarto momento) dos elementos da matriz
MathMoments	Calcula os primeiros 4 momentos (média, dispersão, obliquidade, curtose) dos elementos da matriz
MathMedian	Calcula a mediana dos elementos da matriz
MathStandardDeviation	Calcula o desvio padrão dos elementos da matriz
MathAverageDeviation	Calcula o desvio médio dos elementos da matriz

© 2000-2024, MetaQuotes Ltd.

MathMean

Calcula a média (primeiro momento) dos elementos da matriz. Equivalente a mean() no R.

```
double  MathMean(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para calcular a média.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Média dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

MathVariance

Calcula a dispersão (segundo momento) dos elementos da matriz. Equivalente a var() no R.

```
double  MathVariance(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Dispersão de elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

MathSkewness

Calcula a obliquidade (terceiro momento) dos elementos da matriz. Equivalente a skewness() no R (biblioteca e1071).

```
double  MathSkewness(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Obliquidade dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

MathKurtosis

Calcula a curtose (quarto momento) dos elementos da matriz. Equivalente a kurtosis() no R (biblioteca e1071).

```
double  MathKurtosis(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Curtose dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

Observação

O cálculo da curtose é realizado com respeito à distribuição normal (excess kurtosis=kurtosis-3), ou seja, a curtose de uma distribuição normal é igual a zero.

Ele será positivo, se o pico da distribuição estiver perto da expectativa matemática, ou agudo e negativo, se a parte superior for plana.

© 2000-2024, MetaQuotes Ltd.

MathMoments

Calcula os primeiros 4 momentos (média, dispersão, obliquidade, curtose) dos elementos da matriz.

```
double  MathMoments(
   const double&  array[],              // matriz com dados
   double&        mean,                 // média (primeiro momento)
   double&        variance,             // dispersão (segundo momento)
   double&        skewness,             // coeficiente de obliquidade (terceiro momer
   double&        kurtosis,             // coeficiente de curtose (quarto momento)
   const int      start=0,              // índice inicial
   const int      count=WHOLE_ARRAY     // número de elementos
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

mean

[out] Variável para a média (1 momento)

variance

[out] Variável para a dispersão (2 momento)

skewness

[out] Variável para a obliquidade (3 momento)

kurtosis

[out] Variável para a curtose (4 momento)

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

Observação

O cálculo da curtose é realizado com respeito à distribuição normal (excess kurtosis=kurtosis-3), ou seja, a curtose de uma distribuição normal é igual a zero.

Ele será positivo, se o pico da distribuição estiver perto da expectativa matemática, ou agudo e negativo, se a parte superior for plana.

© 2000-2024, MetaQuotes Ltd.

MathMedian

Calcula a mediana dos elementos da matriz. Equivalente a median() no R.

```
double  MathMedian(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Mediana dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

MathStandardDeviation

Calcula o desvio padrão dos elementos da matriz. Equivalente a sd() no R.

```
double  MathStandardDeviation(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Desvio padrão dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

MathAverageDeviation

Calcula o desvio médio dos elementos da matriz. Equivalente a aad() no R.

```
double  MathAverageDeviation(
   const double&  array[]                // matriz com dados
   );
```

Parâmetros

array

[in] Matriz com os dados para o cálculo.

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Desvio médio dos elementos da matriz. Em caso de erro, retorna NaN (não é um número).

© 2000-2024, MetaQuotes Ltd.

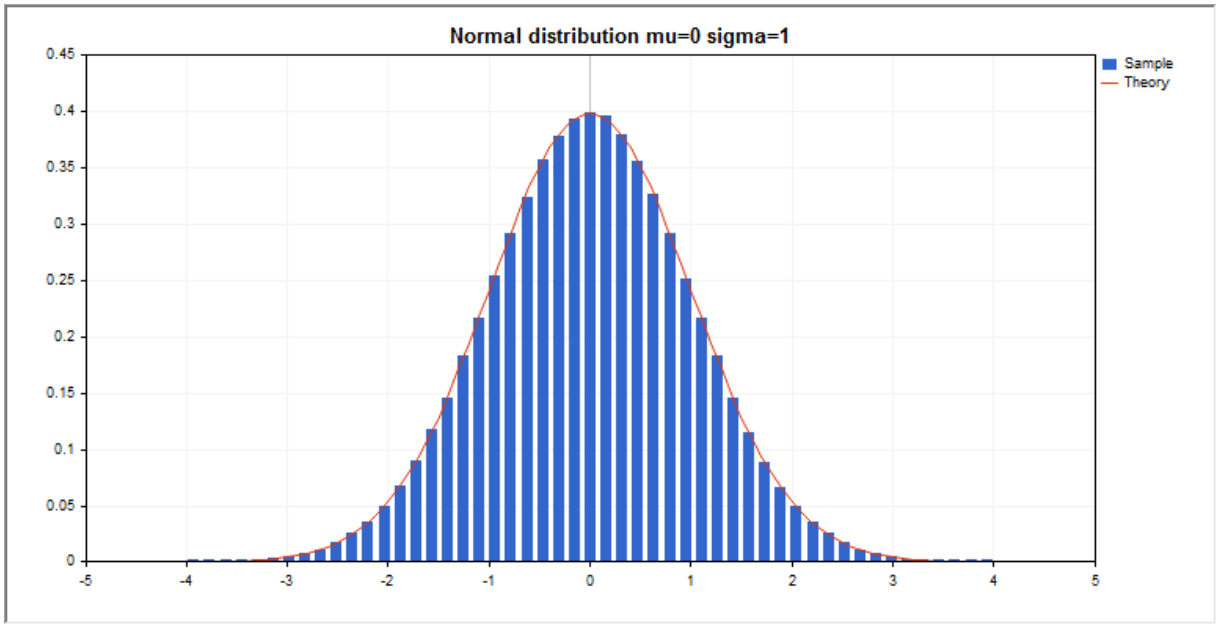
Distribuição normal

Esta seção apresenta as funções para trabalhar com a distribuição normal. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição normal. A distribuição é descrita pela seguinte fórmula:

$$f_{Normal}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Onde:

- x — valor da variável aleatória
- μ — valor esperado
- σ — desvio padrão



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNormal	Calcula a função densidade de probabilidade da distribuição normal
MathCumulativeDistributionNormal	Calcula o valor da função de distribuição normal de probabilidade
MathQuantileNormal	Calcula o inverso da função de distribuição normal para a probabilidade definida
MathRandomNormal	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição normal
MathMomentsNormal	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição normal

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mean_value=0;  // valor esperado (mean)
input double std_dev=1;     // desvio padrão (standard deviation)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição normal
   MathRandomNormal(mean_value,std_dev,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Normal distribution mu=%G sigma=%G",mean_value
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- plot all curves
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNormal

Calcula a função densidade de probabilidade da distribuição normal com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double MathProbabilityDensityNormal(
   const double x,              // valor da variável aleatória
   const double mu,             // parâmetro da distribuição mean (valor esperado)
   const double sigma,          // parâmetro da distribuição sigma (desvio padrão)
   const bool   log_mode,       // cálculo do logaritmo do valor
   int&         error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição normal com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double MathProbabilityDensityNormal(
   const double x,              // valor da variável aleatória
   const double mu,             // parâmetro da distribuição mean (valor esperado)
   const double sigma,          // parâmetro da distribuição sigma (desvio padrão)
   int&         error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição normal com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dnorm() no R.

```
bool MathProbabilityDensityNormal(
   const double& x[],           // matriz com valores da variável aleatória
   const double  mu,            // parâmetro da distribuição mean (valor esperado)
   const double  sigma,         // parâmetro da distribuição sigma (desvio padrão)
   const bool    log_mode,      // cálculo do logaritmo do valor
   double&       result[]       // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição normal com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool MathProbabilityDensityNormal(
   const double& x[],           // matriz com valores da variável aleatória
   const double  mu,            // parâmetro da distribuição mean (valor esperado)
   const double  sigma,         // parâmetro da distribuição sigma (desvio padrão)
   double&       result[]       // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

mu

[in] Parâmetro da distribuição mean (valor esperado).

sigma

[in] Parâmetro da distribuição sigma (desvio padrão).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz para obter os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNormal

Calcula o valor da função de distribuição normal de probabilidade com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   const bool    tail,           // sinalizador para o cálculo da cauda (tail)
   const bool    log_mode,       // cálculo do logaritmo do valor
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o valor da função de distribuição normal de probabilidade com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o valor da função de distribuição normal de probabilidade com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dnorm() no R.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   const bool    tail,           // sinalizador para o cálculo da cauda (tail)
   const bool    log_mode,       // cálculo do logaritmo do valor
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula o valor da função de distribuição normal de probabilidade com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

mu

[in] Parâmetro da distribuição mean (valor esperado).

sigma

[in] Parâmetro da distribuição sigma (desvio padrão).

tail

[in] Sinalizador de cálculo. Se tail=true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz para obter os valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNormal

Calcula para uma determinada probabilidade *probability* o inverso da função de distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valor da probabilidade da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   const bool    tail,           // sinalizador para o cálculo da cauda (tail)
   const bool    log_mode,       // cálculo do logaritmo do valor
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula para uma determinada probabilidade *probability* o inverso da função de distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna NaN.

```
double  MathQuantileNormal(
   const double  probability,    // valor da probabilidade da variável aleatória
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula para a matriz de valores de probabilidade *probability[]* o inverso da função de distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna false. Equivalente a qnorm() no R.

```
bool  MathQuantileNormal(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   const bool    tail,           // sinalizador para o cálculo da cauda (tail)
   const bool    log_mode,       // cálculo do logaritmo do valor
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula para a matriz de valores de probabilidade *probability[]* o inverso da função de distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna false.

```
bool  MathQuantileNormal(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

mu

[in] Parâmetro da distribuição mean (valor esperado).

sigma

[in] Parâmetro da distribuição sigma (desvio padrão).

tail

[in] Sinalizador de cálculo. Se for false, o cálculo será realizado para uma probabilidade de 1.0 - probability

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz para obter quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNormal

Gera um número pseudo-aleatório de acordo com a distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna NaN.

```
double  MathRandomNormal(
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios de acordo com a distribuição normal com parâmetros mu e sigma. Em caso de erro, retorna false. Equivalente a rnorm() no R.

```
bool  MathRandomNormal(
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   const int     data_count,     // número de valores necessários
   double&       result[]        // matriz para obter números pseudo-aleatórios
   );
```

Parâmetros

mu

[in] Parâmetro da distribuição mean (valor esperado).

sigma

[in] Parâmetro da distribuição sigma (desvio padrão).

data_count

[in] Número de valores pseudo-aleatórios que é preciso obter.

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNormal

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição normal.

```
double  MathMomentsNormal(
   const double  mu,             // valor esperado
   const double  sigma,          // desvio padrão
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para registro do código de erro
   );
```

Parâmetros

mu

[in] Parâmetro da distribuição mean (valor esperado).

sigma

[in] Parâmetro da distribuição sigma (desvio padrão).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

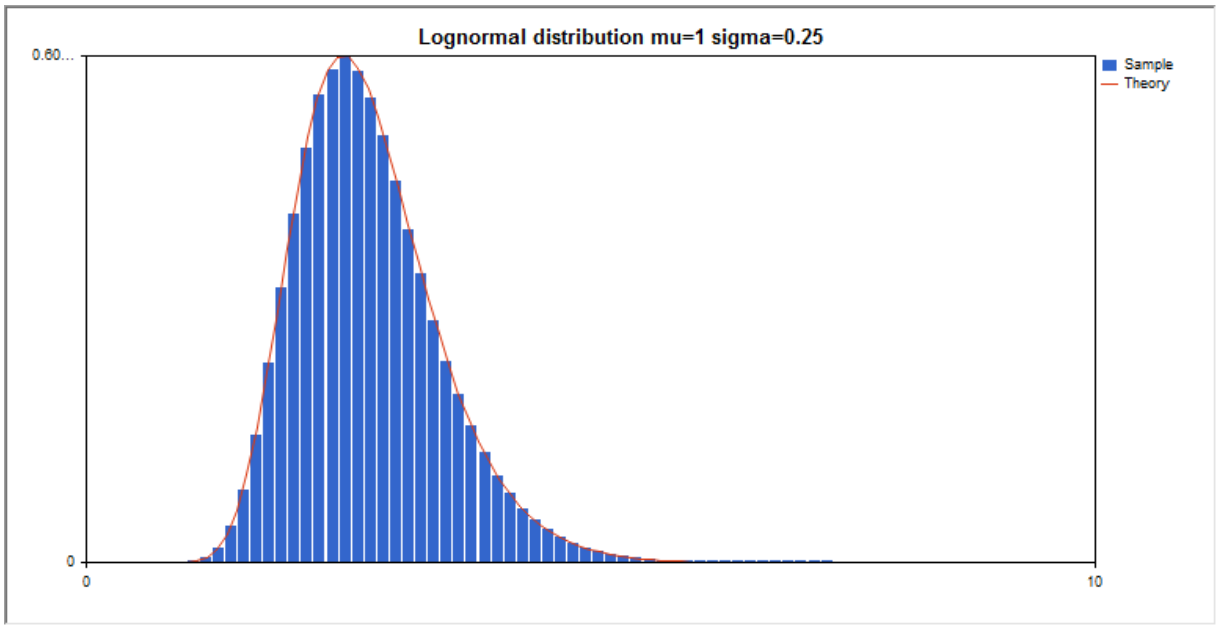
Distribuição log-normal

Esta seção apresenta as funções para trabalhar com a distribuição log-normal. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios distribuídos de acordo com a distribuição log-normal. A distribuição log-normal é descrita pela seguinte fórmula:

$$f_{Lognormal}(x|\mu,\sigma)=\frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

Onde:

- x — valor da variável aleatória
- μ — logaritmo do valor esperado
- σ — logaritmo do desvio padrão



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityLognormal	Calcula a densidade de probabilidade da distribuição log-normal
MathCumulativeDistributionLognormal	Calcula o valor da função distribuição log-normal da probabilidade
MathQuantileLognormal	Calcula o inverso da função de distribuição log-normal para a probabilidade definida
MathRandomLognormal	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição log-normal
MathMomentsLognormal	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição log-normal

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mean_value=1.0;  // logaritmo do valor esperado (log mean)
input double std_dev=0.25;    // logaritmo do desvio padrão (log standard deviation)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição log-normal
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Lognormal distribution mu=%G sigma=%G",mean_va
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityLognormal

Calcula a densidade de probabilidade da distribuição log-normal com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a densidade de probabilidade da distribuição log-normal com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição log-normal com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna NaN. Equivalente a dlnorm() no R.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // matriz com valores da variável aleatória
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool     log_mode,       // cálculo de logaritmo do valor, se log_mode=true,
   double&        result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição log-normal com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],            // matriz com valores da variável aleatória
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   double&        result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

mu

[in] Logaritmo do valor esperado (log_mean).

sigma

[in] Logaritmo do desvio padrão (log standard deviation).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para obter os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

Calcula a distribuição log-normal de probabilidade com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool    tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, i
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a distribuição log-normal de probabilidade com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // valor da variável aleatória
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a distribuição log-normal das probabilidades com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a plnorm() no R.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],            // matriz com valores da variável aleatória
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool     tail,           // sinalizador de cálculo, se for true, é calculada
   const bool     log_mode,       // calcula o logaritmo do valor, se log_mode=true, e
   double&        result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição log-normal das probabilidades com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],            // matriz com valores da variável aleatória
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   double&        result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

mu

[in] Logaritmo do valor esperado (log_mean).

sigma

[in] Logaritmo do desvio padrão (log standard deviation).

tail

[in] Sinalizador de cálculo, se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para obter os valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileLognormal

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a probabilidade probability. Em caso de erro, retorna NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a probabilidade probability. Em caso de erro, retorna NaN.

```
double  MathQuantileLognormal(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a matriz de valores da probabilidade probability[]. Em caso de erro, retorna false. Equivalente a qlnorm() no R.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // matriz com valores de probabilidade da variável a
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   const bool     tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool     log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&        result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a matriz de valores da probabilidade probability[]. Em caso de erro, retorna false.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // matriz com valores de probabilidade da variável a
   const double   mu,             // logaritmo do valor esperado (log mean)
   const double   sigma,          // logaritmo do desvio padrão (log standard deviatio
   double&        result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade de aparecimento da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

mu

[in] Logaritmo do valor esperado (log_mean).

sigma

[in] Logaritmo do desvio padrão (log standard deviation).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomLognormal

Gera um número pseudo-aleatório de acordo com a distribuição log-normal com parâmetros mu e sigma. Em caso de erro, retorna NaN.

```
double  MathRandomLognormal(
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios de acordo com a distribuição log-normal com parâmetros mu e sigma. Em caso de erro, retorna false. Equivalente a rlnorm() no R.

```
double  MathRandomLognormal(
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóri
   );
```

Parâmetros

mu

[in] Logaritmo do valor esperado (log_mean).

sigma

[in] Logaritmo do desvio padrão (log standard deviation).

data_count

[in] Número de dados necessários.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz com os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsLognormal

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição log-normal. Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

```
double  MathMomentsLognormal(
   const double  mu,             // logaritmo do valor esperado (log mean)
   const double  sigma,          // logaritmo do desvio padrão (log standard deviatio
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para registro do código de erro
   );
```

Parâmetros

mu

[in] Logaritmo do valor esperado (log_mean).

sigma

[in] Logaritmo do desvio padrão (log standard deviation).

mean

[in] Variável para o valor médio.

variance

[out] Variável para a dispersão.

skewness

[out] Variável para a obliquidade.

kurtosis

[out] Variável para a curtose.

error_code

[out] Variável para registro do código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

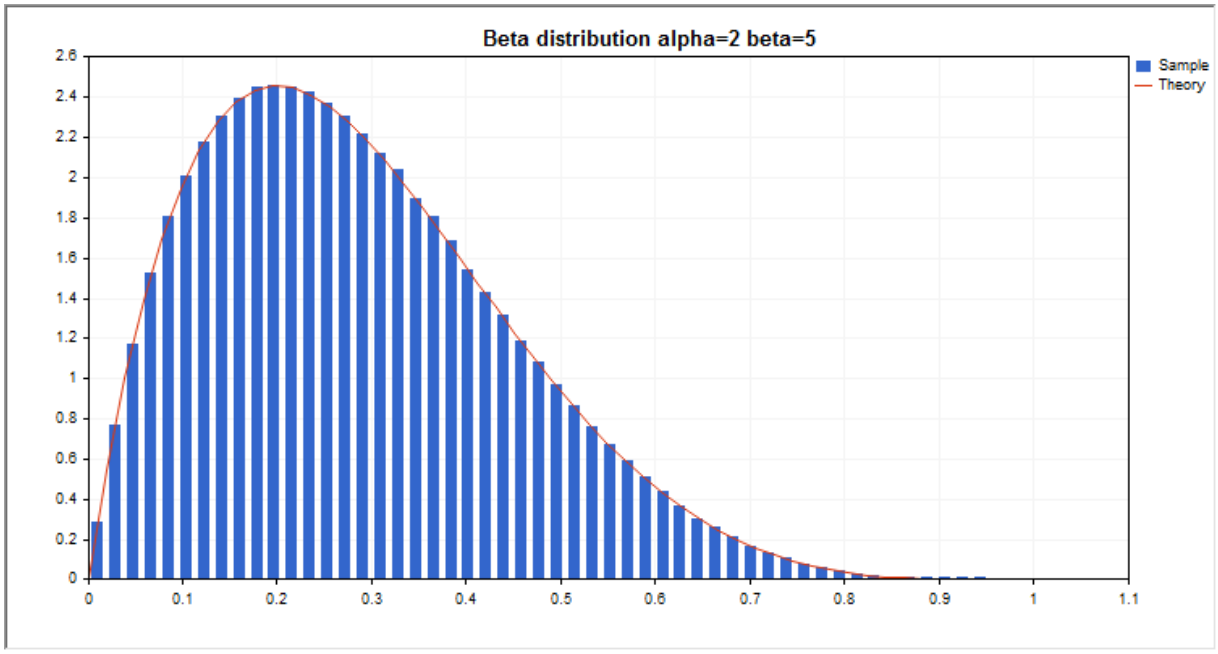
Distribuição beta

Esta seção apresenta as funções para trabalhar com a distribuição beta. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição correspondente. A distribuição beta é descrita pela seguinte fórmula:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

Onde:

- x — valor da variável aleatória
- a — primeiro parâmetro de distribuição beta
- b — segundo parâmetro de distribuição beta



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityBeta	Calcula a densidade de probabilidade da distribuição beta
MathCumulativeDistributionBeta	Calcula o valor da função distribuição beta de probabilidades
MathQuantileBeta	Calcula o inverso da função de distribuição beta para a probabilidade especificada
MathRandomBeta	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição beta
MathMomentsBeta	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição beta

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=2;   // primeiro parâmetro de distribuição beta (shape1)
input double beta=5;    // segundo parâmetro de distribuição beta (shape2)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição beta
   MathRandomBeta(alpha,beta,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Beta distribution alpha=%G beta=%G",alpha,beta
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityBeta

Calcula a função densidade de probabilidade da distribuição beta com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valor da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição beta com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityBeta(
   const double  x,              // valor da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de densidade de probabilidade da distribuição beta com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dbeta() no R.

```
bool  MathProbabilityDensityBeta(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // primeiro parâmetro da distribuição beta (shape1)
   const double  b,               // segundo parâmetro da distribuição beta (shape2)
   const bool    log_mode,        // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Calcula a função de densidade de probabilidade da distribuição beta com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityBeta(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // primeiro parâmetro da distribuição beta (shape1)
   const double  b,               // segundo parâmetro da distribuição beta (shape2)
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

a

[in] Primeiro parâmetro da distribuição beta (shape 1).

b

[in] Segundo parâmetro da distribuição beta (shape 2).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para o valor da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBeta

Calcula a distribuição de probabilidade beta com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const bool    tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade beta com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade beta com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pbeta() no R.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const bool    tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidade beta com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

a

[in] Primeiro parâmetro da distribuição beta (shape 1).

b

[in] Segundo parâmetro da distribuição beta (shape 2).

tail

[in] Sinalizador de cálculo, se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileBeta

Calcula - para a probabilidade *probability* - o inverso da função de distribuição beta com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula - para a probabilidade *probability* - o inverso da função de distribuição beta com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathQuantileBeta(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula - para a matriz dos valores - a probabilidade *probability[]* o inverso da função de distribuição beta com os parâmetros a e b. Em caso de erro, retorna false. Equivalente a qbeta() no R.

```
double  MathQuantileBeta(
   const double&  probability[], // matriz com valores de probabilidade da variável a
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula - para a matriz dos valores - a probabilidade *probability[]* o inverso da função de distribuição beta com os parâmetros a e b. Em caso de erro, retorna false.

```
bool  MathQuantileBeta(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Primeiro parâmetro da distribuição beta (shape1).

b

[in] Segundo parâmetro da distribuição beta (shape2).

tail

[in] Sinalizador de cálculo, se lower_tail=false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomBeta

Gera um número pseudo-aleatório distribuído de acordo com a distribuição beta com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathRandomBeta(
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição beta com parâmetros a e b. Em caso de erro, retorna false. Equivalente a rbeta() no R.

```
bool  MathRandomBeta(
   const double  a,               // primeiro parâmetro da distribuição beta (shape1)
   const double  b,               // segundo parâmetro da distribuição beta (shape2)
   const int     data_count,      // número de dados necessários
   double&       result[]         // matriz para obter números pseudo-aleatórios
   );
```

Parâmetros

a

[in] primeiro parâmetro da distribuição beta (shape1)

b

[in] Segundo parâmetro da distribuição beta (shape2).

data_count

[in] Número de valores pseudo-aleatórios que é preciso obter.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsBeta

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição beta.

```
double  MathMomentsBeta(
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Primeiro parâmetro da distribuição beta (shape1).

b

[in] Segundo parâmetro da distribuição beta (shape2).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

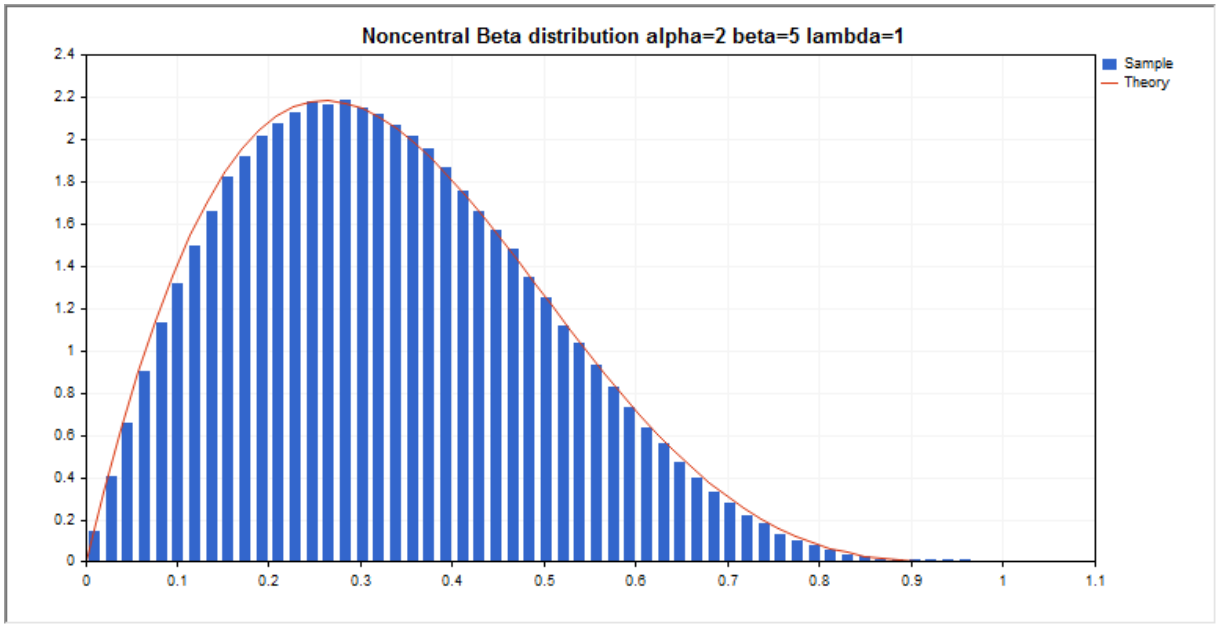
Distribuição beta não-central

Esta seção apresenta as funções para trabalhar com a distribuição beta não-central. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição correspondente. A distribuição beta não-central é descrita pela seguinte fórmula:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^{r}}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

Onde:

- x — valor da variável aleatória
- a — primeiro parâmetro de distribuição beta
- b — segundo parâmetro de distribuição beta
- λ — parâmetro de não-centralidade



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNoncentralBeta	Calcula a densidade de probabilidade da distribuição beta não-central
MathCumulativeDistributionNoncentralBeta	Calcula o valor da função distribuição beta não-central de probabilidades
MathQuantileNoncentralBeta	Calcula o inverso da função de distribuição beta não-central para a probabilidade especificada
MathRandomNoncentralBeta	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição beta não-central

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathMomentsNoncentralBeta	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição beta não-central

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=2;    // primeiro parâmetro de distribuição beta (shape1)
input double b_par=5;    // segundo parâmetro de distribuição beta (shape2)
input double l_par=1;    // parâmetro de não-centralidade (lambda)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=53;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição beta não-central
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2024, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral Beta distribution alpha=%G beta=%G
                          a_par,b_par,l_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

Calcula a função densidade de probabilidade da distribuição beta não-central com parâmetros a, b e lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, i
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição beta não-central com parâmetros a, b e lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição beta não-central com parâmetros a, b e lambda para a matriz das variáveis aleatórias x[].. Em caso de erro, retorna false. Equivalente a dbeta() no R.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Calcula a função densidade de probabilidade da distribuição beta não-central com parâmetros a, b e lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Parâmetros

© 2000-2024, MetaQuotes Ltd.

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

a

[in] Primeiro parâmetro da distribuição beta (shape 1).

b

[in] Segundo parâmetro da distribuição beta (shape 2).

lambda

[in] Parâmetro de não-centralidade

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

Calcula a distribuição de probabilidade beta não-central com parâmetros a, b e lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   const bool    tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade beta não-central com parâmetros a, b e lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade beta não-central com parâmetros a, b e lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pbeta() no R.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],           // matriz com valores da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   const bool    tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]       // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidade beta não-central com parâmetros a, b e lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],           // matriz com valores da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   double&       result[]       // matriz para valores da função de probabilidade
   );
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

a

[in] Primeiro parâmetro da distribuição beta (shape 1).

b

[in] Segundo parâmetro da distribuição beta (shape 2).

lambda

[in] Parâmetro de não-centralidade

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralBeta

Calcula a função inversa da distribuição de probabilidades beta não-central com parâmetros a, b e lambda para a probabilidade de aparecimento do valor da variável aleatória *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função inversa da distribuição de probabilidades beta não-central com parâmetros a, b e lambda para a probabilidade de aparecimento do valor da variável aleatória *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função inversa da distribuição de probabilidades beta não-central com parâmetros a, b e lambda para a matriz de valores da probabilidade probability[]. Em caso de erro, retorna false. Equivalente a qbeta() no R.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula a função inversa da distribuição de probabilidades beta não-central com parâmetros a, b e lambda para a matriz de valores da probabilidade probability[]. Em caso de erro, retorna false.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
```

© 2000-2024, MetaQuotes Ltd.

```
    double&        result[]        // matriz com valores de quantis
    );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Primeiro parâmetro da distribuição beta (shape1).

b

[in] Segundo parâmetro da distribuição beta (shape2).

lambda

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralBeta

Gera um número pseudo-aleatório distribuído de acordo com a distribuição beta não-central com parâmetros a. b e lambda. Em caso de erro, retorna NaN.

```
double  MathRandomNoncentralBeta(
   const double  a,             // primeiro parâmetro da distribuição beta (shape1)
   const double  b,             // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,        // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição beta não-central com parâmetros a, b e lambda. Em caso de erro, retorna false. Equivalente a rbeta() no R.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz para obter números pseudo-aleatórios
   );
```

Parâmetros

a

[in] primeiro parâmetro da distribuição beta (shape1)

b

[in] Segundo parâmetro da distribuição beta (shape2).

lambda

[in] Parâmetro de não-centralidade

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralBeta

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição beta não-central com parâmetros a, b e lambda

```
double  MathMomentsNoncentralBeta(
   const double  a,              // primeiro parâmetro da distribuição beta (shape1)
   const double  b,              // segundo parâmetro da distribuição beta (shape2)
   const double  lambda,         // parâmetro de não-centralidade
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Primeiro parâmetro da distribuição beta (shape1).

b

[in] Segundo parâmetro da distribuição beta (shape2).

lambda

[in] Parâmetro de não-centralidade

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

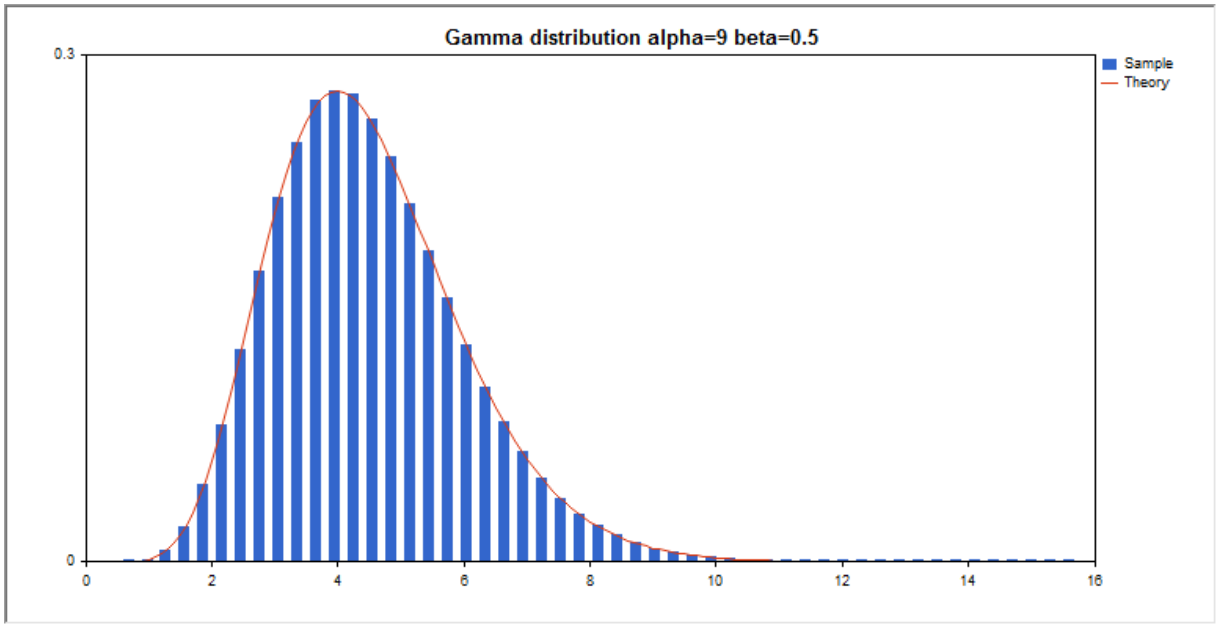
Distribuição gama

Esta seção apresenta as funções para trabalhar com a distribuição gama. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição correspondente. A distribuição gama é descrita pela seguinte fórmula:

$$f_{Gamma}(x \mid a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

Onde:

- x — valor da variável aleatória
- a — primeiro parâmetro de distribuição
- b — segundo parâmetro de distribuição



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityGamma	Calcula a densidade de probabilidade da distribuição gama
MathCumulativeDistributionGamma	Calcula o valor da função distribuição gama de probabilidades
MathQuantileGamma	Calcula o inverso da função de distribuição gama para a probabilidade especificada
MathRandomGamma	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição gama
MathMomentsGamma	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição gama

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double alpha=9;   // primeiro parâmetro de distribuição gama (shape)
input double beta=0.5;  // segundo parâmetro de distribuição gama (shape)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição gama
   MathRandomGamma(alpha,beta,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%G",alpha,bet
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityGamma

Calcula a função densidade de probabilidade da distribuição gama com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valor da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição gama com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityGamma(
   const double  x,              // valor da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de densidade de probabilidade da distribuição gama com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dgamma() no R.

```
bool  MathProbabilityDensityGamma(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // primeiro parâmetro da distribuição (shape)
   const double  b,               // segundo parâmetro da distribuição (scale)
   const bool    log_mode,        // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

«Calcula a função de densidade de probabilidade da distribuição gama com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityGamma(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // primeiro parâmetro da distribuição (shape)
   const double  b,               // segundo parâmetro da distribuição (scale)
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

a

[in] Primeiro parâmetro da distribuição (shape).

b

[in] Segundo parâmetro da distribuição (scale).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionGamma

Calcula a distribuição gama de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição (shape)
   const double  b,             // segundo parâmetro da distribuição (scale)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição gama de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionGamma(
   const double  x,             // valor da variável aleatória
   const double  a,             // primeiro parâmetro da distribuição (shape)
   const double  b,             // segundo parâmetro da distribuição (scale)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição gama de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pgamma() no R.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição gama de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

a

[in] Primeiro parâmetro da distribuição (shape).

b

[in] Segundo parâmetro da distribuição (scale)

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileGamma

Calcula o inverso da função de distribuição gama com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição gama com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileGamma(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição gama com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qgamma() no R.

```
double  MathQuantileGamma(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição gama com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileGamma(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Primeiro parâmetro da distribuição (shape).

b

[in] Segundo parâmetro da distribuição (scale).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomGamma

Gera um número pseudo-aleatório distribuído de acordo com a distribuição gama com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathRandomGamma(
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Gera um número pseudo-aleatório distribuído de acordo com a distribuição gama com parâmetros a e b. Em caso de erro, retorna false. Equivalente a rgamma() no R.

```
bool  MathRandomGamma(
   const double  a,               // primeiro parâmetro da distribuição (shape)
   const double  b,               // segundo parâmetro da distribuição (scale)
   const int     data_count,      // número de dados necessários
   double&       result[]         // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

a

[in] Primeiro parâmetro da distribuição (shape).

b

[in] Segundo parâmetro da distribuição (scale).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsGamma

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição gama com parâmetros a, b.

```
double  MathMomentsGamma(
   const double  a,              // primeiro parâmetro da distribuição (shape)
   const double  b,              // segundo parâmetro da distribuição (scale)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Primeiro parâmetro da distribuição (shape).

b

[in] Segundo parâmetro da distribuição (scale).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

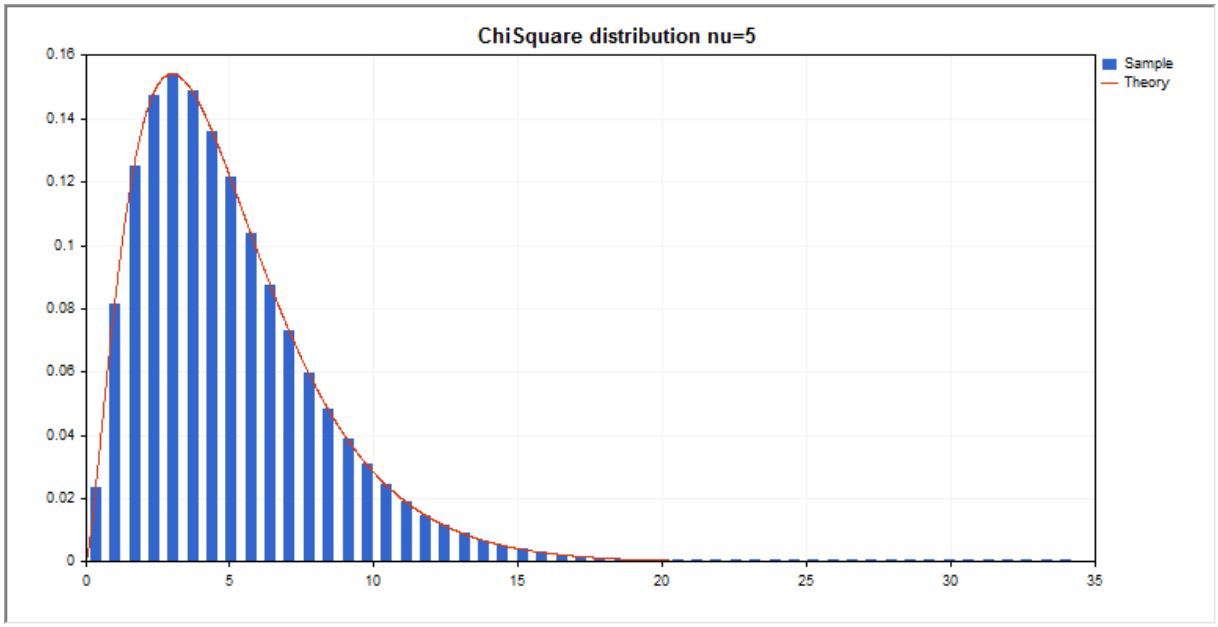
Qui-quadrado

Esta seção contém funções para trabalhar com a distribuição qui-quadrado. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição correspondente. A distribuição qui-quadrado é descrita pela seguinte fórmula:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

Onde:

- x — valor da variável aleatória
- ν — número de graus de liberdade



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityChiSquare	Calcula a densidade de probabilidade da distribuição qui-quadrado
MathCumulativeDistributionChiSquare	Calcula o valor da função distribuição de probabilidades qui-quadrado
MathQuantileChiSquare	Calcula o inverso da função de distribuição qui-quadrado para a probabilidade especificada
MathRandomChiSquare	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição qui-quadrado
MathMomentsChiSquare	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição qui-quadrado

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=5;    // número de graus de libertade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a seleção a partir da distribuição qui-quadrado
   MathRandomChiSquare(nu_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("ChiSquare distribution nu=%G ",nu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
```

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

Calcula a função densidade de probabilidade da distribuição qui-quadrado com parâmetro nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado com parâmetro nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado com parâmetro nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dchisq() no R.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],             // matriz com valores da variável aleatória
   const double  nu,              // parâmetro de distribuição (número de graus de libe
   const bool    log_mode,        // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado com parâmetro nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],             // matriz com valores da variável aleatória
   const double  nu,              // parâmetro de distribuição (número de graus de libe
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade)

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

Calcula a distribuição de probabilidades qui-quadrado com parâmetros nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades qui-quadrado com parâmetros nu para a variável aleatória x. Em caso de erro, retorna NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades qui-quadrado com parâmetros nu para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pchisq() no R.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidades qui-quadrado com parâmetros nu para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileChiSquare

Calcula o inverso da função de distribuição de probabilidades qui-quadrado para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de probabilidades qui-quadrado para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileChiSquare(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de probabilidades qui-quadrado para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qchisq() no R.

```
double  MathQuantileChiSquare(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição de probabilidades qui-quadrado para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomChiSquare

Gera um número pseudo-aleatório distribuído de acordo com a distribuição qui-quadrado com parâmetro nu. Em caso de erro, retorna NaN.

```
double  MathRandomChiSquare(
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   int&          error_code     // variável para registro do código de erro
   );
```

Gera um número pseudo-aleatório distribuído de acordo com a distribuição qui-quadrado com parâmetro nu. Em caso de erro, retorna false. Equivalente a rchisq() no R.

```
bool  MathRandomChiSquare(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsChiSquare

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição qui-quadrado com parâmetro nu.

```
double  MathMomentsChiSquare(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

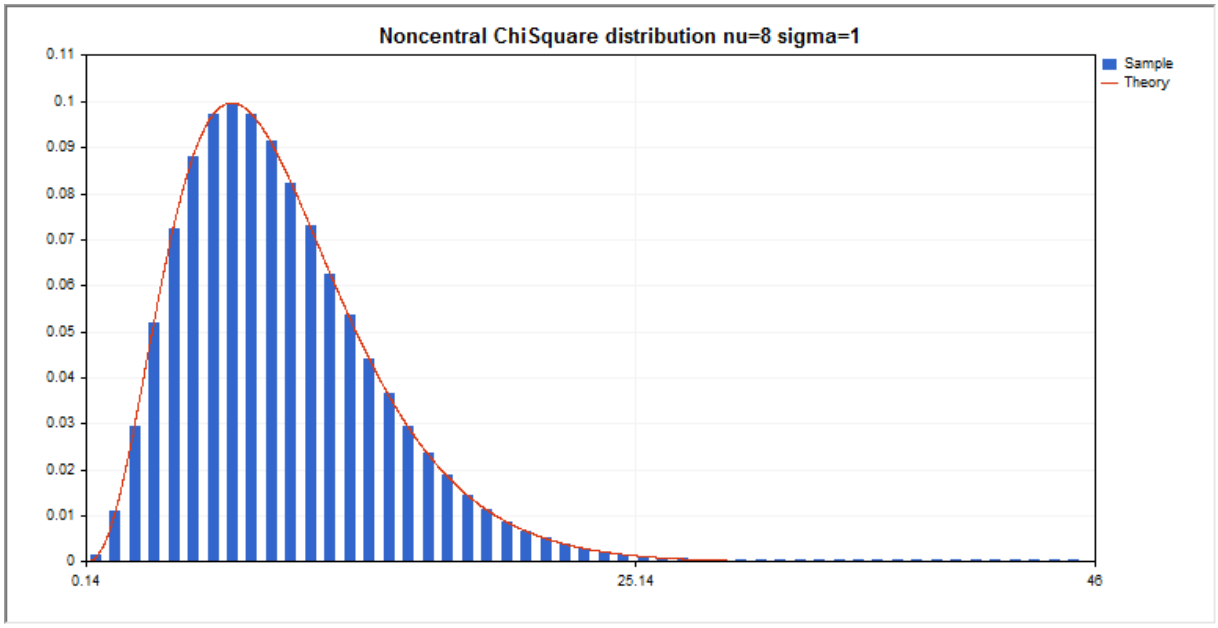
Qui-quadrado não-central

Esta seção contém funções para trabalhar com a distribuição qui-quadrado não-central. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição correspondente. A distribuição qui-quadrado não-central é descrita pela seguinte fórmula:

$$f_{NoncentralChiSquare}(x|\nu,\sigma)=\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)}x^{\frac{\nu}{2}-1}e^{-\frac{(x+\sigma)}{2}}\sum_{r=0}^{\infty}\frac{(\lambda x)^r}{(2r)!}\frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

Onde:

- x — valor da variável aleatória
- ν — número de graus de liberdade
- σ — parâmetro de não-centralidade



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNoncentralChiSquare	Calcula a densidade de probabilidade da distribuição qui-quadrado não-central
MathCumulativeDistributionNoncentralChiSquare	Calcula o valor da função distribuição de probabilidades qui-quadrado não-central
MathQuantileNoncentralChiSquare	Calcula o inverso da função de distribuição qui-quadrado não-central para a probabilidade especificada
MathRandomNoncentralChiSquare	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição qui-quadrado não-central

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathMomentsNoncentralChiSquare	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição qui-quadrado não-central

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=8;    // número de graus de liberdade
input double si_par=1;    // parâmetro de não-centralidade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a seleção a partir da distribuição qui-quadrado não-central
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral ChiSquare distribution nu=%G sigma=
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2024, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

Calcula a função densidade de probabilidade da distribuição qui-quadrado não-central com parâmetros nu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  sigma,         // parâmetro de não-centralidade
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado não-central com parâmetros nu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  sigma,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado não-central com parâmetros nu e sigma para matiz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dchisq() no R.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição qui-quadrado não-central com parâmetro nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

Calcula a distribuição de probabilidades qui-quadrado não-central com parâmetros nu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  sigma,         // parâmetro de não-centralidade
   const bool    tail,          // sinalizador de cálculo, se lower_tail=true, é calcu
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades qui-quadrado não-central com parâmetros nu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  sigma,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades qui-quadrado não-central com parâmetros nu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pchisq() no R.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se lower_tail=true, é calc
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidades qui-quadrado não-central com parâmetros nu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

Calcula o inverso da função de distribuição qui-quadrado não-central com parâmetros mu e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição qui-quadrado não-central com parâmetros mu e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição qui-quadrado não-central com parâmetros mu e sigma para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qchisq() no R.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // matriz com valores de probabilidade da variável a
   const double  nu,              // parâmetro de distribuição (número de graus de lib
   const double  sigma,           // parâmetro de não-centralidade
   const bool    tail,            // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,        // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]         // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição de probabilidades qui-quadrado não-central para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

Gera um número pseudo-aleatório distribuído de acordo com a distribuição qui-quadrado não-central com parâmetros nu e sigma. Em caso de erro, retorna NaN.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  sigma,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição qui-quadrado não-central com parâmetros nu e sigma. Em caso de erro, retorna false. Equivalente a rchisq() no R.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição qui-quadrado não-central com parâmetros nu e sigma.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  sigma,          // parâmetro de não-centralidade
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

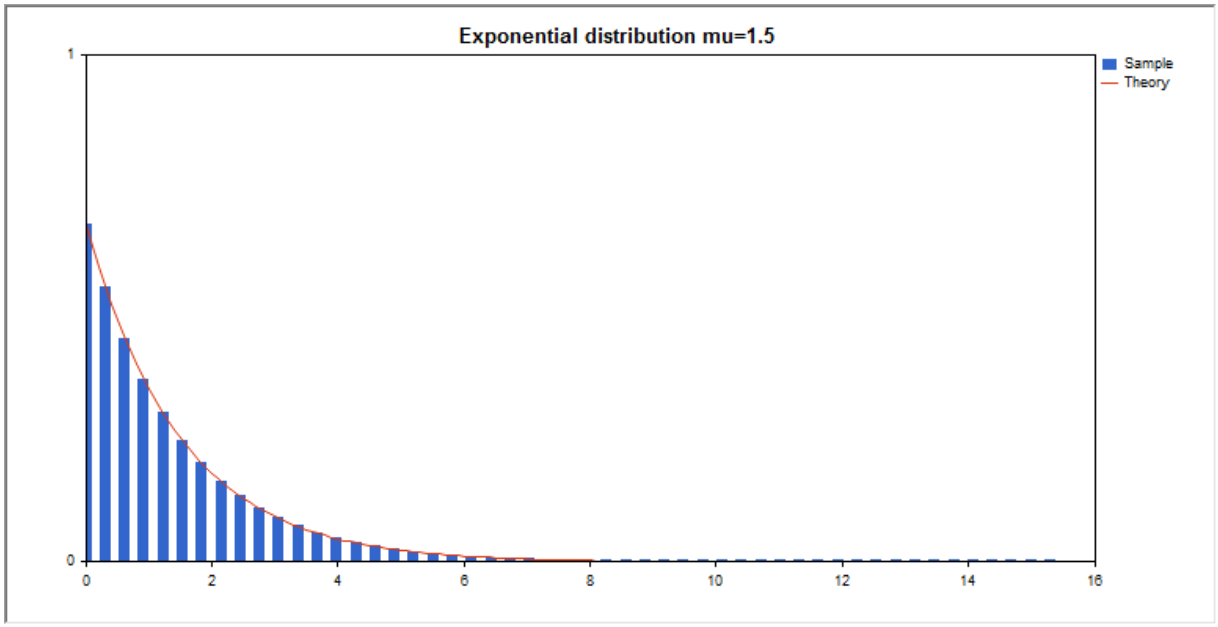
Exponencial

Esta seção apresenta as funções para trabalhar com a distribuição exponencial. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição exponencial. A distribuição exponencial é descrita pela seguinte fórmula:

$$f_{Exponential}(x \mid \mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

Onde:

- x — valor da variável aleatória
- μ — valor esperado



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityExponential	Calcula a densidade de probabilidade da distribuição exponencial
MathCumulativeDistributionExponential	Calcula o valor da função distribuição exponencial da probabilidade
MathQuantileExponential	Calcula o inverso da função de distribuição exponencial para a probabilidade especificada
MathRandomExponential	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição exponencial
MathMomentsExponential	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição exponencial

Exemplo:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2024, MetaQuotes Ltd.

```
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mu_par=1.5;    // número de graus de liberdade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição exponencial
   MathRandomExponential(mu_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
```

© 2000-2024, MetaQuotes Ltd.

```
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityExponential

Calcula a função densidade de probabilidade da distribuição exponencial com parâmetro mu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valor da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, r
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição exponencial com parâmetro mu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityExponential(
   const double  x,              // valor da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição exponencial com parâmetro mu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dexp() no R.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Calcula a função densidade de probabilidade da distribuição exponencial com parâmetro mu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

mu

[in] Parâmetro de distribuição (valor esperado)

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionExponential

Calcula a distribuição exponencial de probabilidade com parâmetro mu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro de distribuição (valor esperado)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição exponencial de probabilidade com parâmetro mu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionExponential(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro de distribuição (valor esperado)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição exponencial de probabilidade com parâmetro mu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pexp() no R.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição exponencial de probabilidade com parâmetro mu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro de distribuição (valor esperado)
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

mu

[in] Parâmetro de distribuição (valor esperado).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileExponential

Calcula o inverso da função de distribuição exponencial com parâmetro mu para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição exponencial com parâmetro mu para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileExponential(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // parâmetro de distribuição (valor esperado)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição exponencial com parâmetro mu para a probabilidade para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qexp() no R.

```
double  MathQuantileExponential(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição exponencial com parâmetro mu para a probabilidade para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileExponential(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  mu,             // parâmetro de distribuição (valor esperado)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

mu

© 2000-2024, MetaQuotes Ltd.

[in] Parâmetro de distribuição (valor esperado).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomExponential

Gera um número pseudo-aleatório distribuído de acordo com a distribuição exponencial com parâmetro mu. Em caso de erro, retorna NaN.

```
double  MathRandomExponential(
   const double  mu,            // parâmetro de distribuição (valor esperado)
   int&          error_code     // variável para registro do código de erro
   );
```

Gera um números pseudo-aleatórios distribuídos de acordo com a distribuição exponencial com parâmetro mu. Em caso de erro, retorna false. Equivalente a rexp() no R.

```
bool  MathRandomExponential(
   const double  mu,             // parâmetro de distribuição (valor esperado)
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

mu

[in] Parâmetro de distribuição (valor esperado).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsExponential

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição exponencial com parâmetro mu.

```
double  MathMomentsExponential(
   const double  mu,             // parâmetro de distribuição (valor esperado)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

mu

[in] Parâmetro de distribuição (valor esperado).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

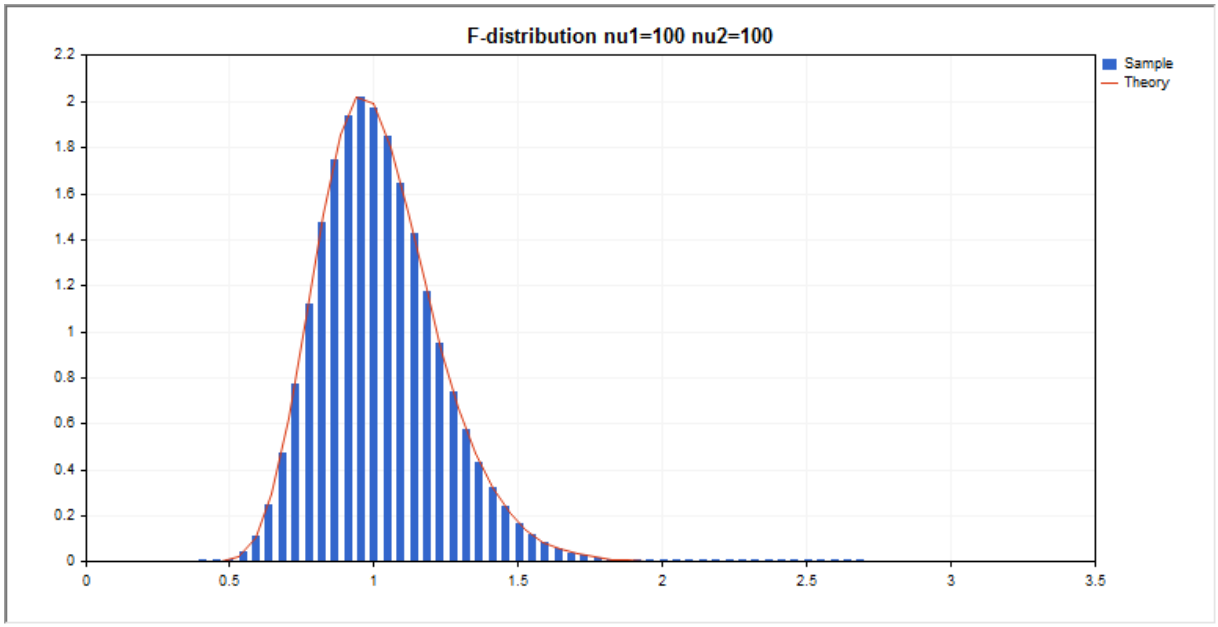
Distribuição F de Fisher-Snedecor

Esta seção apresenta as funções para trabalhar com a distribuição F de Fisher-Snedecor. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição F. A distribuição F de Fisher-Snedecor é descrita pela seguinte fórmula:

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

Onde:

- x — valor da variável aleatória
- ν1 — primeiro parâmetro de distribuição (número de graus de liberdade)
- ν2 — segundo parâmetro de distribuição (número de graus de liberdade)



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityF	Calcula a densidade de probabilidade da distribuição F de Fisher-Snedecor
MathCumulativeDistributionF	Calcula o valor da função distribuição F de Fisher-Snedecor de probabilidades
MathQuantileF	Calcula o inverso da função de distribuição F de Fisher-Snedecor para a probabilidade especificada
MathRandomF	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição F de Fisher-Snedecor

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathMomentsF	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição F de Fisher-Snedecor

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_1=100;    // primeiro número de graus de liberdade
input double nu_2=100;    // segundo número de graus de liberdade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição F de Fisher-Snedecor
   MathRandomF(nu_1,nu_2,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("F-distribution nu1=%G nu2=%G",nu_1,nu_2));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityF

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valor da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de graus
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityF(
   const double  x,              // valor da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de graus
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a df() no R.

```
bool  MathProbabilityDensityF(
   const double& x[],             // matriz com valores da variável aleatória
   const double  nu1,             // primeiro parâmetro da distribuição (número de grau
   const double  nu2,             // segundo parâmetro da distribuição (número de graus
   const bool    log_mode,        // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityF(
   const double& x[],             // matriz com valores da variável aleatória
   const double  nu1,             // primeiro parâmetro da distribuição (número de grau
   const double  nu2,             // segundo parâmetro da distribuição (número de graus
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionF

Calcula a distribuição de probabilidades F de Fisher-Snedecor com parâmetros nu1 e nu2 para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // cálculo de logaritmo do valor, se log_mode=true, é
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades F de Fisher-Snedecor com parâmetros nu1 e nu2 para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pf() no R.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, é
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidade F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileF

Calcula o inverso da função de distribuição de probabilidades F de Fisher-Snedecor com parâmetros nu1 e nu2 para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de probabilidades F de Fisher-Snedecor com parâmetros nu1 e nu2 para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qf() no R.

```
double  MathQuantileF(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2 para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileF(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

tail

[in] Sinalizador de cálculo, se lower_tail=false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomF

Gera um número pseudo-aleatório distribuído de acordo com a distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2. Em caso de erro, retorna NaN.

```
double  MathRandomF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   int&          error_code     // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2. Em caso de erro, retorna false. Equivalente a rf() no R.

```
bool  MathRandomF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsF

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição F de Fisher-Snedecor com parâmetros nu1 e nu2

```
double  MathMomentsF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

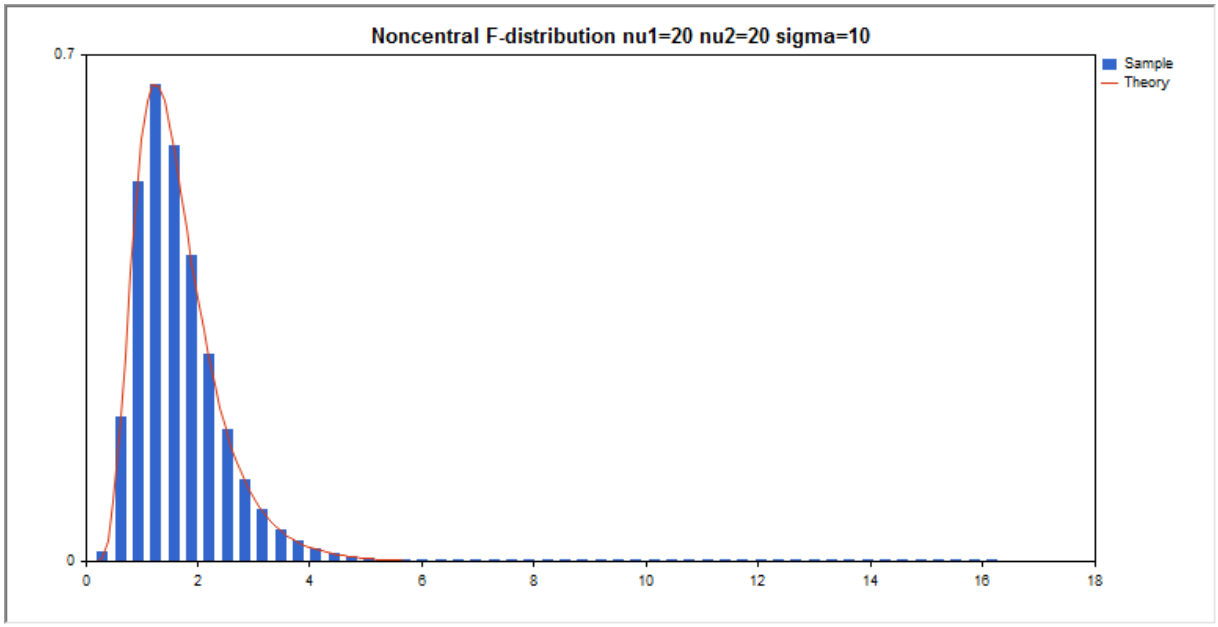
Distribuição F de Fisher-Snedecor não-central

Esta seção apresenta as funções para trabalhar com a distribuição F de Fisher-Snedecor não-central. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição F não central. A distribuição F de Fisher-Snedecor não-central é descrita pela seguinte fórmula:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma)=e^{-\frac{\sigma}{2}}\sum_{r=0}^{\infty}\frac{1}{r!}\left(\frac{\sigma}{2}\right)^r\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r}\frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

Onde:

- x — valor da variável aleatória
- ν1 — primeiro parâmetro de distribuição (número de graus de liberdade)
- ν2 — segundo parâmetro de distribuição (número de graus de liberdade)
- σ — parâmetro de não-centralidade



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNoncentralF	Calcula a densidade de probabilidade da distribuição F de Fisher-Snedecor não-central
MathCumulativeDistributionNoncentralF	Calcula o valor da função distribuição F de Fisher-Snedecor não-central de probabilidades
MathQuantileNoncentralF	Calcula o inverso da função de distribuição F de Fisher-Snedecor não-central para a probabilidade especificada

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathRandomNoncentralF	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição F de Fisher-Snedecor não-central
MathMomentsNoncentralF	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição F de Fisher-Snedecor não-central

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_1=20;    // primeiro número de graus de liberdade
input double nu_2=20;    // segundo número de graus de liberdade
input double sig=10;     // parâmetro de não-centralidade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição F de Fisher-Snedecor não-central
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral F-distribution nu1=%G nu2=%G sigma=
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2024, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a df() no R.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

© 2000-2024, MetaQuotes Ltd.

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

Calcula a distribuição de probabilidade F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,             // valor da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de graus
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidade F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pf() no R.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],           // matriz com valores da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de grau
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]       // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidade F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],           // matriz com valores da variável aleatória
   const double  nu1,           // primeiro parâmetro da distribuição (número de grau
   const double  nu2,           // segundo parâmetro da distribuição (número de graus
   const double  sigma,         // parâmetro de não-centralidade
   double&       result[]       // matriz para valores da função de probabilidade
   );
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileF

Calcula o inverso da função de distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileF(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qf() no R.

```
double  MathQuantileF(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileF(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu1,            // primeiro parâmetro da distribuição (número de gra
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

© 2000-2024, MetaQuotes Ltd.

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralF

Gera um número pseudo-aleatório distribuído de acordo com a distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma. Em caso de erro, retorna NaN.

```
double  MathRandomNoncentralF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma. Em caso de erro, retorna false. Equivalente a rf() no R.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralF

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição F de Fisher-Snedecor não-central com parâmetros nu1, nu2 e sigma.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // primeiro parâmetro da distribuição (número de grau
   const double  nu2,            // segundo parâmetro da distribuição (número de graus
   const double  sigma,          // parâmetro de não-centralidade
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu1

[in] Primeiro parâmetro de distribuição (número de graus de liberdade).

nu2

[in] Segundo parâmetro de distribuição (número de graus de liberdade).

sigma

[in] Parâmetro de não-centralidade.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

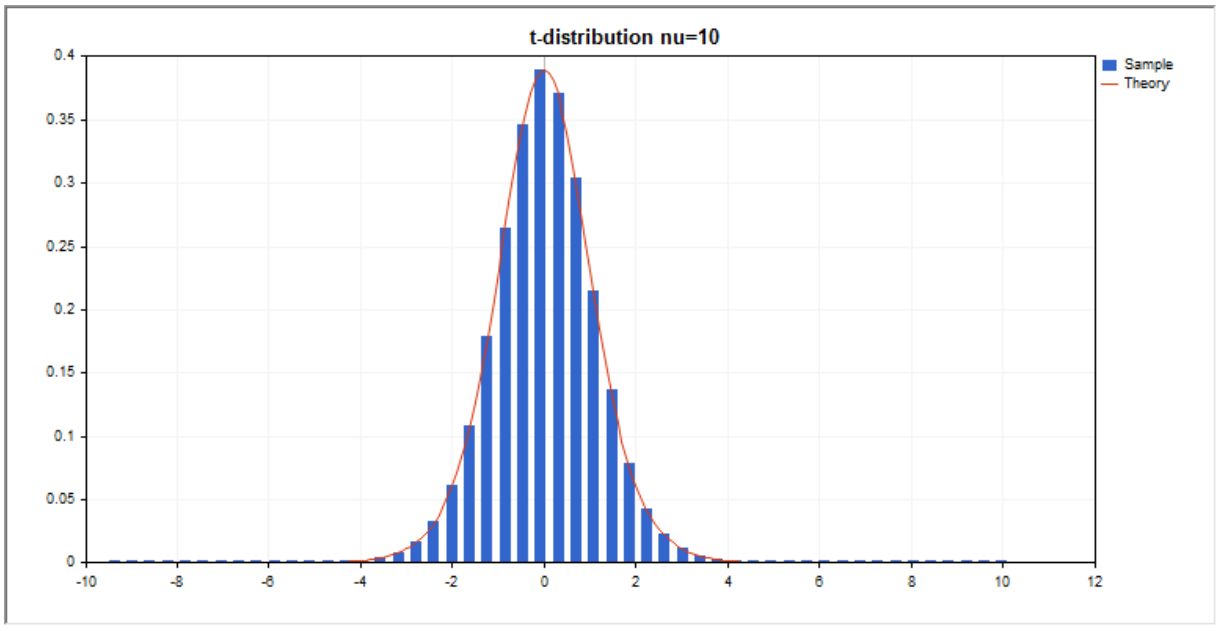
Distribuição t de Student

Esta seção apresenta as funções para trabalhar com a distribuição t de Student. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição t de Student. A distribuição t de Student é descrita pela seguinte fórmula:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

Onde:

- x — valor da variável aleatória
- ν — parâmetro de distribuição (número de graus de liberdade)



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityT	Calcula a densidade de probabilidade da distribuição t de Student
MathCumulativeDistributionT	Calcula o valor da função distribuição t de Student de probabilidades
MathQuantileT	Calcula o inverso da função de distribuição t de Student para a probabilidade especificada
MathRandomT	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição t de Student
MathMomentsT	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição t de Student

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=10;    // número de graus de liberdade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição F de Fisher-Snedecor
   MathRandomT(nu_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2024, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("t-distribution nu=%G",nu_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
```

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityT

Calcula a função densidade de probabilidade da distribuição t de Student com parâmetro nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student com parâmetro nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityT(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student com parâmetro nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dt() no R.

```
bool  MathProbabilityDensityT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student com parâmetro nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionT

Calcula a distribuição t de Student com parâmetros nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionT(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição t de Student com parâmetros nu para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionT(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição t de Student com parâmetros nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pt() no R.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição t de Student com parâmetros nu para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileT

Calcula o inverso da função de distribuição t de Student com parâmetro nu para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileT(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição t de Student com parâmetro nu para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileT(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição t de Student com parâmetro nu para a matriz de valores das probabilidades *probability[]*. Em caso de erro, retorna false. Equivalente a qt() no R.

```
double  MathQuantileT(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição t de Student com parâmetro nu para a matriz de valores das probabilidades *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileT(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomT

Gera um número pseudo-aleatório distribuído de acordo com a distribuição t de Student com parâmetro nu. Em caso de erro, retorna NaN.

```
double  MathRandomT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição t de Student com parâmetro nu. Em caso de erro, retorna false. Equivalente a rt() no R.

```
bool  MathRandomT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsT

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição t de Student com parâmetro nu.

```
double  MathMomentsT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

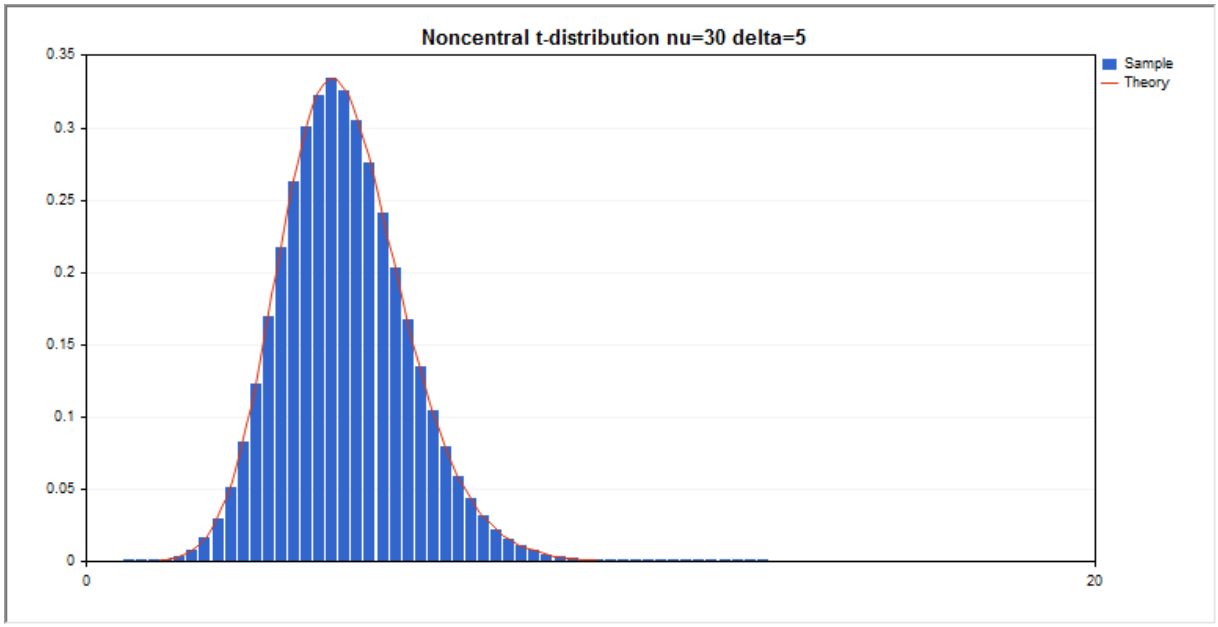
Distribuição t de Student

Esta seção apresenta as funções para trabalhar com a distribuição t de Student não-central. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição t de Student não-central. A distribuição t de Student não-central é descrita pela seguinte fórmula:

$$f_{NoncentralT}(x \mid \nu, \delta) = \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu + x^2\right)^{\frac{\nu+1}{2}}} \sum_{r=0}^{\infty} \frac{(x\delta)^r}{r!} \left(\frac{2}{\nu + x^2}\right)^{\frac{r}{2}} \Gamma\left(\frac{\nu + r + 1}{2}\right)$$

Onde:

- x — valor da variável aleatória
- ν — parâmetro de distribuição (número de graus de liberdade)
- σ — parâmetro de não-centralidade



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNoncentralT	Calcula a densidade de probabilidade da distribuição t de Student não-central
MathCumulativeDistributionNoncentralT	Calcula o valor da função distribuição t de Student não-central de probabilidades
MathQuantileNoncentralT	Calcula o inverso da função de distribuição t de Student não-central para a probabilidade especificada
MathRandomNoncentralT	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição t de Student não-central

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathMomentsNoncentralT	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição t de Student não-central

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double nu_par=30;      // número de graus de liberdade
input double delta_par=5;    // parâmetro de não-centralidade
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição F de Fisher-Snedecor não-central
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2024, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral t-distribution nu=%G delta=%G",nu_p
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

Calcula a função densidade de probabilidade da distribuição t de Student não-central com parâmetros nu e delta para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para o código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student não-central com parâmetros nu e delta para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // valor da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   int&          error_code      // variável para o código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student não-central com parâmetros nu e delta para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dt() no R.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição t de Student não-central com parâmetros nu e delta para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

delta

[in] Parâmetro de não-centralidade.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

Calcula a distribuição de probabilidades t de Student não-central com parâmetros nu e delta para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  delta,         // parâmetro de não-centralidade
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades t de Student não-central com parâmetros nu e delta para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,             // valor da variável aleatória
   const double  nu,            // parâmetro de distribuição (número de graus de libe
   const double  delta,         // parâmetro de não-centralidade
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades t de Student não-central com parâmetros nu e delta para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pt() no R.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidades t de Student não-central com parâmetros nu e delta para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // matriz com valores da variável aleatória
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

delta

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNoncentralT

Calcula o inverso da função de distribuição t de Student não-central com parâmetros nu e delta para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se lower_tail=false, o cá
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição t de Student não-central com parâmetros nu e delta para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNoncentralT(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição t de Student não-central com parâmetros nu e delta para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qt() no R.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const bool    tail,           // sinalizador de cálculo, se lower_tail=false, o cá
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição t de Student não-central com parâmetros nu e delta para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

delta

[in] Parâmetro de não-centralidade.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNoncentralT

Gera um número pseudo-aleatório distribuído de acordo com a distribuição t de Student não-central com parâmetros nu e delta. Em caso de erro, retorna NaN.

```
double  MathRandomNoncentralT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição t de Student não-central com parâmetros nu e delta. Em caso de erro, retorna false. Equivalente a rt() no R.

```
bool  MathRandomNoncentralT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

delta

[in] Parâmetro de não-centralidade.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNoncentralT

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição t de Student não-central com parâmetros nu e delta.

```
double  MathMomentsNoncentralT(
   const double  nu,             // parâmetro de distribuição (número de graus de libe
   const double  delta,          // parâmetro de não-centralidade
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

nu

[in] Parâmetro de distribuição (número de graus de liberdade).

delta

[in] Parâmetro de não-centralidade.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

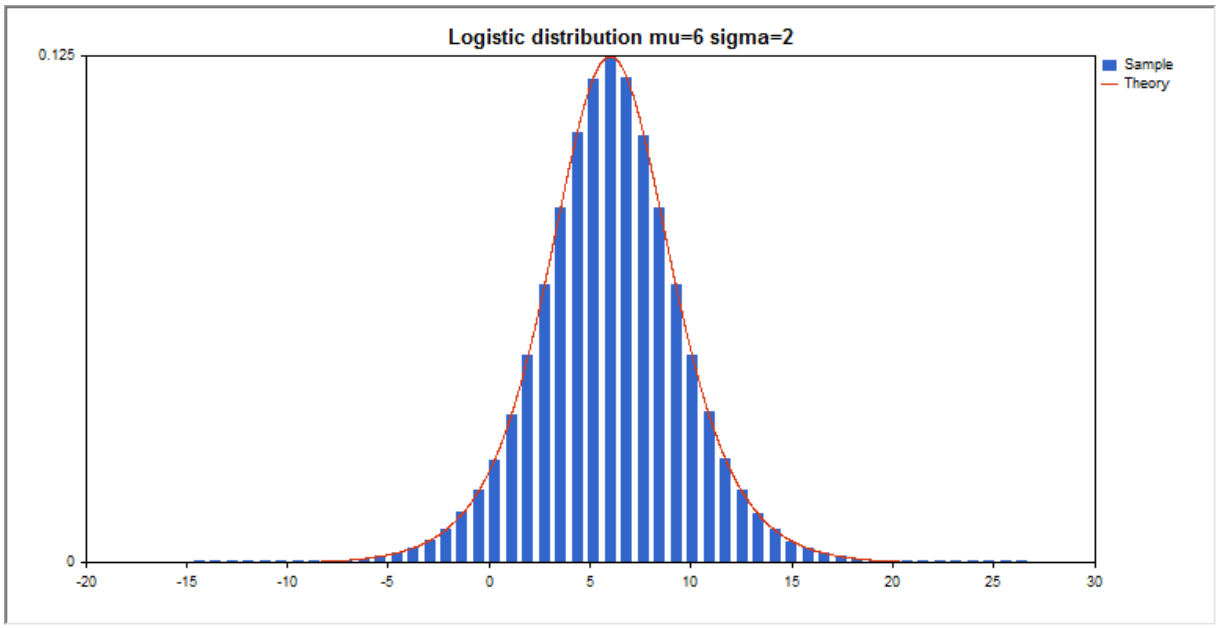
Distribuição logística

Esta seção apresenta as funções para trabalhar com a distribuição logística. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição logística. A distribuição logística é descrita pela seguinte fórmula:

$$f_{Logistic}(x|\mu,\sigma)=\frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

Onde:

- x — valor da variável aleatória
- μ — parâmetro de distribuição mean
- σ — parâmetro de distribuição scale



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityLogistic	Calcula a densidade de probabilidade da distribuição logística
MathCumulativeDistributionLogistic	Calcula o valor da função distribuição logística de probabilidades
MathQuantileLogistic	Calcula o inverso da função de distribuição logística para a probabilidade especificada
MathRandomLogistic	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição logística
MathMomentsLogistic	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição logística

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double mu_par=6;        // parâmetro de distribuição mean
input double sigma_par=2;     // parâmetro de distribuição scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição logística
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Logistic distribution mu=%G sigma=%G",mu_par,s
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityLogistic

Calcula a função densidade de probabilidade da distribuição logística com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro da distribuição mean
   const double  sigma,         // parâmetro da distribuição scale
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição logística com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityLogistic(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro da distribuição mean
   const double  sigma,         // parâmetro da distribuição scale
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição logística com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dlogis() no R.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição logística com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

mu

[in] Parâmetro da distribuição mean.

sigma

[in] Parâmetro da distribuição scale.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

Calcula a distribuição logística de probabilidades com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro da distribuição mean
   const double  sigma,         // parâmetro da distribuição scale
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição logística de probabilidades com parâmetros mu e sigma para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionLogistic(
   const double  x,             // valor da variável aleatória
   const double  mu,            // parâmetro da distribuição mean
   const double  sigma,         // parâmetro da distribuição scale
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição logística das probabilidades com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição logística das probabilidades com parâmetros mu e sigma para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a plogis() no R.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // matriz com valores da variável aleatória
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

mu

[in] Parâmetro da distribuição mean.

sigma

[in] Parâmetro da distribuição scale.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileLogistic

Calcula o inverso da função de distribuição logística com parâmetros mu e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição logística com parâmetros mu e sigma para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileLogistic(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a matriz de valores da probabilidade probability. Em caso de erro, retorna false. Equivalente a qlogis() no R.

```
double  MathQuantileLogistic(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição log-normal com parâmetros mu e sigma para a matriz de valores da probabilidade probability. Em caso de erro, retorna false.

```
bool  MathQuantileLogistic(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

mu

[in] Parâmetro da distribuição mean.

sigma

[in] Parâmetro da distribuição scale.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomLogistic

Gera um número pseudo-aleatório distribuído de acordo com a distribuição logística com parâmetros mu e sigma. Em caso de erro, retorna NaN.

```
double  MathRandomLogistic(
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição logística com parâmetros mu e sigma. Em caso de erro, retorna false. Equivalente a rlogis() no R.

```
bool  MathRandomLogistic(
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

mu

[in] Parâmetro da distribuição mean.

sigma

[in] Parâmetro da distribuição scale.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsLogistic

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição logística com parâmetros mu e sigma.

```
double  MathMomentsLogistic(
   const double  mu,             // parâmetro da distribuição mean
   const double  sigma,          // parâmetro da distribuição scale
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

mu

[in] Parâmetro da distribuição mean.

sigma

[in] Parâmetro da distribuição scale.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

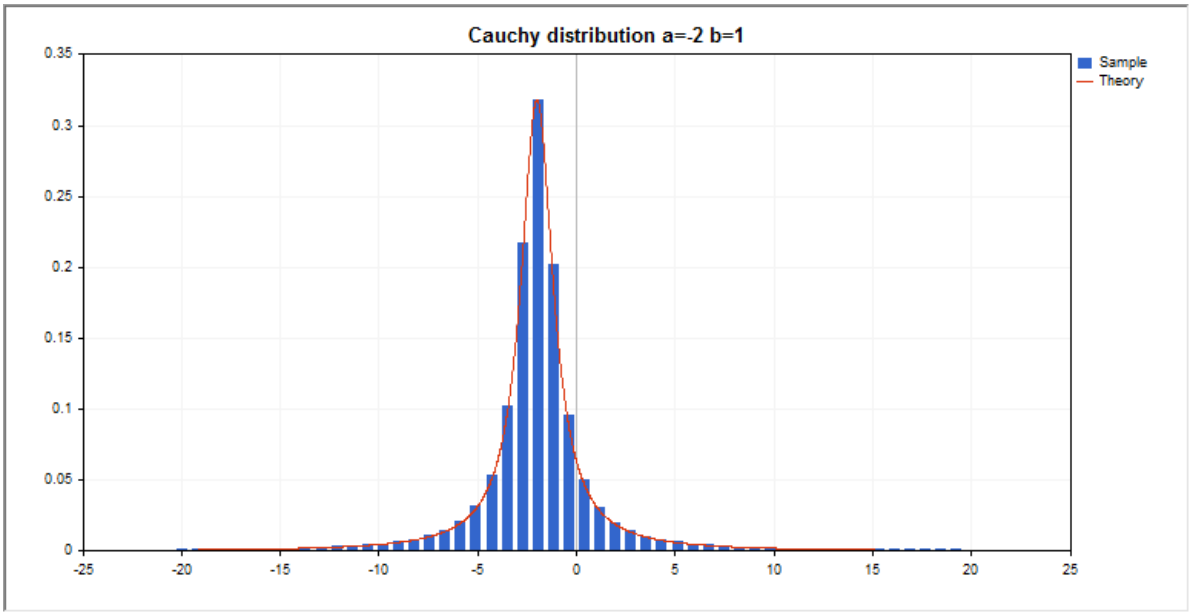
Distribuição de Cauchy

Esta seção contém funções para trabalhar com a distribuição de Cauchy. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição de Cauchy. A distribuição de Cauchy é descrita pela seguinte fórmula:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

Onde:

- x — valor da variável aleatória
- a — parâmetro de distribuição mean
- b — parâmetro de distribuição scale



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityCauchy	Calcula a densidade de probabilidade da distribuição de Cauchy
MathCumulativeDistributionCauchy	Calcula o valor da função distribuição de Cauchy de probabilidades
MathQuantileCauchy	Calcula o inverso da função de distribuição de Cauchy para a probabilidade especificada
MathRandomCauchy	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição de Cauchy
MathMomentsCauchy	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Cauchy

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=-2;      // parâmetro de distribuição mean
input double b_par=1;       // parâmetro de distribuição scale
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição de Cauchy
   MathRandomCauchy(a_par,b_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Cauchy distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1)
      return(false);
   int size=ArraySize(data);
   if(size<cells*10)
      return(false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   Print("min=",minv," max=",maxv);
   minv=-20;
   maxv=20;
   double range=maxv-minv;
   double width=range/cells;
   if(width==0)
      return(false);
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityCauchy

Calcula a função densidade de probabilidade da distribuição de Cauchy com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro da distribuição mean
   const double  b,             // parâmetro da distribuição scale
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição de Cauchy com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityCauchy(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro da distribuição mean
   const double  b,             // parâmetro da distribuição scale
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição de Cauchy com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dcauchy() no R.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição de Cauchy com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

a

[in] Parâmetro da distribuição mean.

b

[in] Parâmetro da distribuição scale.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

Calcula a distribuição de probabilidades de Cauchy com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro da distribuição mean
   const double  b,             // parâmetro da distribuição scale
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades de Cauchy com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionCauchy(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro da distribuição mean
   const double  b,             // parâmetro da distribuição scale
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de probabilidades de Cauchy com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de probabilidades de Cauchy com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a plogis() no R.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

a

[in] Parâmetro da distribuição mean.

b

[in] Parâmetro da distribuição scale.

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileCauchy

Calcula o inverso da função de distribuição de Cauchy com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Cauchy com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileCauchy(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Cauchy com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qcauschy() no R.

```
double  MathQuantileCauchy(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição de Cauchy com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileCauchy(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Parâmetro da distribuição mean.

b

[in] Parâmetro da distribuição scale.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomCauchy

Gera um número pseudo-aleatório distribuído de acordo com a distribuição de Cauchy com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathRandomCauchy(
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição de Cauchy com parâmetros a e b. Em caso de erro, retorna false. Equivalente a rcauchy() no R.

```
bool  MathRandomCauchy(
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

a

[in] Parâmetro da distribuição mean.

b

[in] Parâmetro da distribuição scale.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsCauchy

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Cauchy com parâmetros a, b.

```
double  MathMomentsCauchy(
   const double  a,              // parâmetro da distribuição mean
   const double  b,              // parâmetro da distribuição scale
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Parâmetro da distribuição mean.

b

[in] Parâmetro da distribuição scale.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

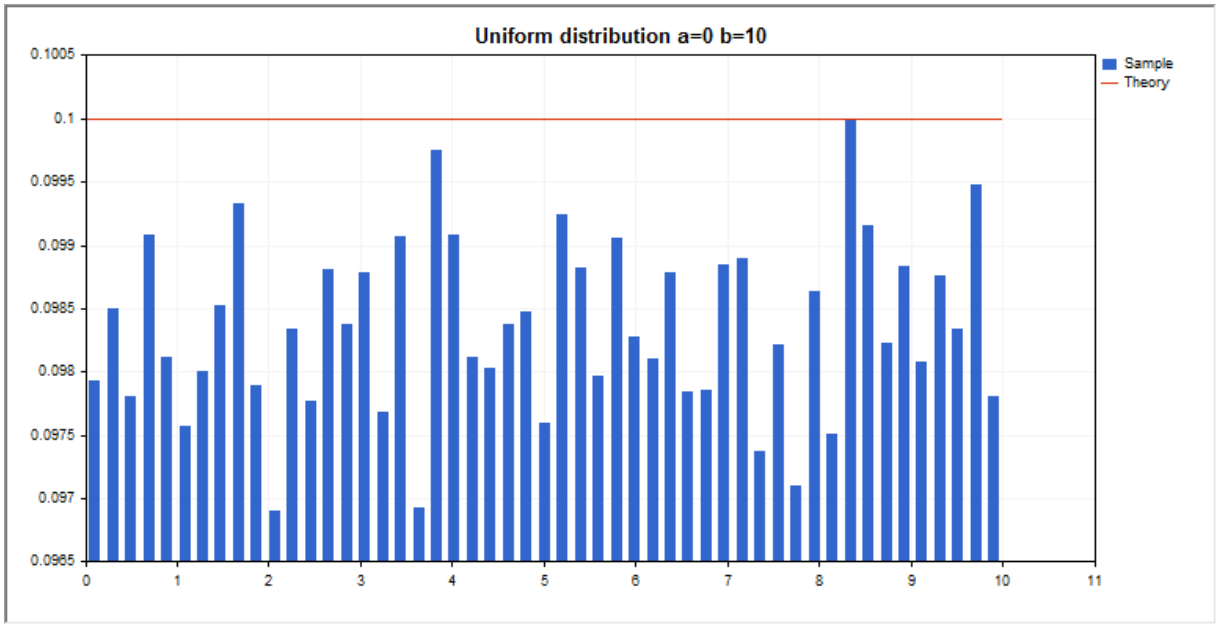
Distribuição uniforme

Esta seção apresenta as funções para trabalhar com a distribuição uniforme. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição uniforme. A distribuição uniforme é descrita pela seguinte fórmula:

$$f_{Uniform}(x|\,a,b) = \frac{1}{b-a}$$

Onde:

- x — valor da variável aleatória
- a — parâmetro de distribuição (limite inferior)
- b — parâmetro de distribuição (limite superior)



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityUniform	Calcula a densidade de probabilidade da distribuição uniforme
MathCumulativeDistributionUniform	Calcula o valor da função distribuição uniforme de probabilidades
MathQuantileUniform	Calcula o inverso da função de distribuição uniforme para a probabilidade especificada
MathRandomUniform	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição uniforme
MathMomentsUniform	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição uniforme

Exemplo:

© 2000-2024, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=0;      // parâmetro de distribuição a (limite inferior)
input double b_par=10;     // parâmetro de distribuição b (limite superior)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição uniforme
   MathRandomUniform(a_par,b_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Uniform distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
```

© 2000-2024, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityUniform

Calcula a função densidade de probabilidade da distribuição uniforme com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição a (limite inferior)
   const double  b,             // parâmetro de distribuição b (limite superior)
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição uniforme com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityUniform(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição a (limite inferior)
   const double  b,             // parâmetro de distribuição b (limite superior)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de densidade de probabilidade da distribuição uniforme com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dunif() no R.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função de densidade de probabilidade da distribuição uniforme com parâmetros a e b para a matriz de variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

a

[in] Parâmetro de distribuição a (limite inferior).

b

[in] Parâmetro de distribuição b (limite superior).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionUniform

Calcula a distribuição uniforme de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição a (limite inferior)
   const double  b,             // parâmetro de distribuição b (limite superior)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição uniforme de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionUniform(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição a (limite inferior)
   const double  b,             // parâmetro de distribuição b (limite superior)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição uniforme de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição uniforme de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a punif() no R.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

a

[in] Parâmetro de distribuição a (limite inferior).

b

[in] Parâmetro de distribuição b (limite superior).

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileUniform

Calcula o inverso da função de distribuição uniforme com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição uniforme com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileUniform(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição uniforme com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qcauschy() no R.

```
double  MathQuantileUniform(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição uniforme com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileUniform(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Parâmetro de distribuição a (limite inferior).

b

[in] Parâmetro de distribuição b (limite superior).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomUniform

Gera um número pseudo-aleatório distribuído de acordo com a distribuição uniforme com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathRandomUniform(
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição uniforme com parâmetros a e b. Em caso de erro, retorna false. Equivalente a runif() no R.

```
bool  MathRandomUniform(
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

a

[in] Parâmetro de distribuição a (limite inferior).

b

[in] Parâmetro de distribuição b (limite superior).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsUniform

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição uniforme com parâmetros a, b.

```
double  MathMomentsUniform(
   const double  a,              // parâmetro de distribuição a (limite inferior)
   const double  b,              // parâmetro de distribuição b (limite superior)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Parâmetro de distribuição a (limite inferior).

b

[in] Parâmetro de distribuição b (limite superior).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

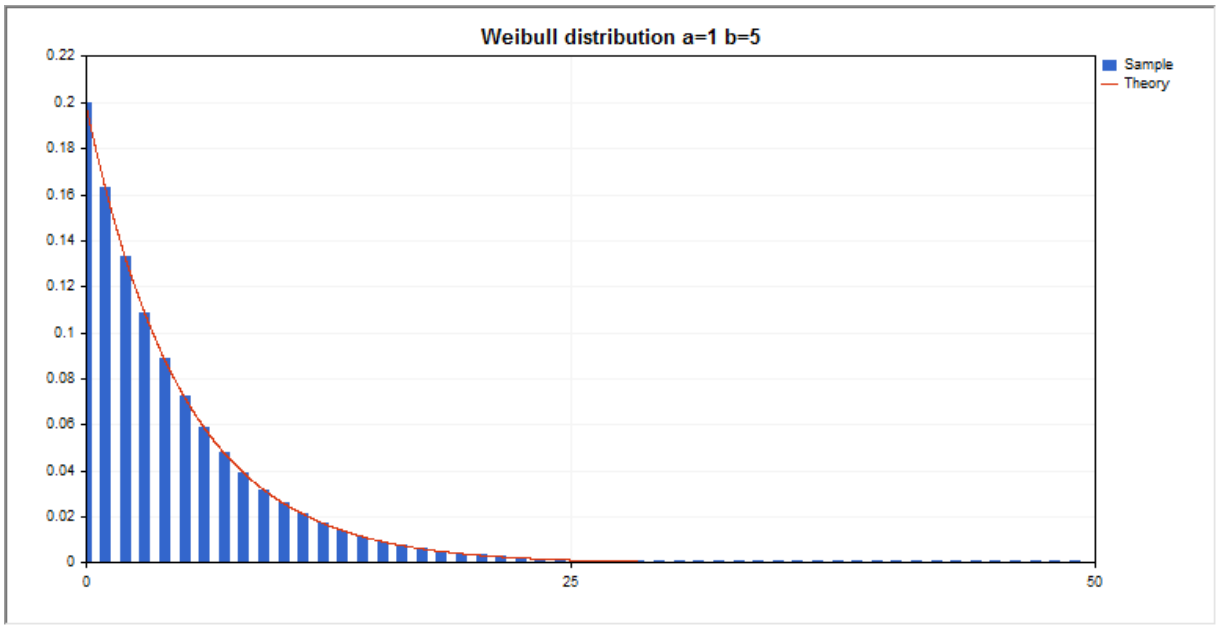
Distribuição de Weibull

Esta seção contém funções para trabalhar com a distribuição de Weibull. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição de Weibull. A distribuição de Weibull é descrita pela seguinte fórmula:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

Onde:

- x — valor da variável aleatória
- a — parâmetro de distribuição (shape)
- b — parâmetro de distribuição (scale)



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityWeibull	Calcula a densidade de probabilidade da distribuição de Weibull
MathCumulativeDistributionWeibull	Calcula o valor da função distribuição de Weibull de probabilidades
MathQuantileWeibull	Calcula o inverso da função de distribuição de Weibull para a probabilidade especificada
MathRandomWeibull	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição de Weibull
MathMomentsWeibull	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Weibull

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double a_par=1;      // parâmetro de distribuição (shape)
input double b_par=5;      // parâmetro de distribuição (scale)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=51;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição de Weibull
   MathRandomWeibull(a_par,b_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
```

© 2000-2024, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Weibull distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityWeibull

Calcula a função densidade de probabilidade da distribuição de Weibull com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valor da variável aleatória
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição de Weibull com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // valor da variável aleatória
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função densidade de probabilidade da distribuição de Weibull com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dweibull() no R.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // parâmetro de distribuição (shape)
   const double  b,               // parâmetro de distribuição (scale)
   const bool    log_mode,        // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Calcula a função densidade de probabilidade da distribuição de Weibull com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],             // matriz com valores da variável aleatória
   const double  a,               // parâmetro de distribuição (shape)
   const double  b,               // parâmetro de distribuição (scale)
   double&       result[]         // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

© 2000-2024, MetaQuotes Ltd.

a

[in] Parâmetro de distribuição (scale).

b

[in] Parâmetro de distribuição (shape).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

Calcula a distribuição de Weibull de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição (shape)
   const double  b,             // parâmetro de distribuição (scale)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de Weibull de probabilidades com parâmetros a e b para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionWeibull(
   const double  x,             // valor da variável aleatória
   const double  a,             // parâmetro de distribuição (shape)
   const double  b,             // parâmetro de distribuição (scale)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição de Weibull de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pweibull() no R.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição de Weibull de probabilidades com parâmetros a e b para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // matriz com valores da variável aleatória
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

a

[in] Parâmetro de distribuição (scale).

b

[in] Parâmetro de distribuição (shape).

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileWeibull

Calcula o inverso da função de distribuição de Weibull com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Weibull com parâmetros a e b para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileWeibull(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Weibull com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qweibull() no R.

```
double  MathQuantileWeibull(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição de Weibull com parâmetros a e b para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileWeibull(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

a

[in] Parâmetro de distribuição (scale).

b

[in] Parâmetro de distribuição (shape).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomWeibull

Gera um número pseudo-aleatório distribuído de acordo com a distribuição de Weibull com parâmetros a e b. Em caso de erro, retorna NaN.

```
double  MathRandomWeibull(
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição de Weibull com parâmetros a e b. Em caso de erro, retorna false. Equivalente a rweibull() no R.

```
bool  MathRandomWeibull(
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

a

[in] Parâmetro de distribuição (scale).

b

[in] Parâmetro de distribuição (shape).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsWeibull

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Weibull com parâmetros a, b.

```
double  MathMomentsWeibull(
   const double  a,              // parâmetro de distribuição (shape)
   const double  b,              // parâmetro de distribuição (scale)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

a

[in] Parâmetro de distribuição (scale).

b

[in] Parâmetro de distribuição (shape).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

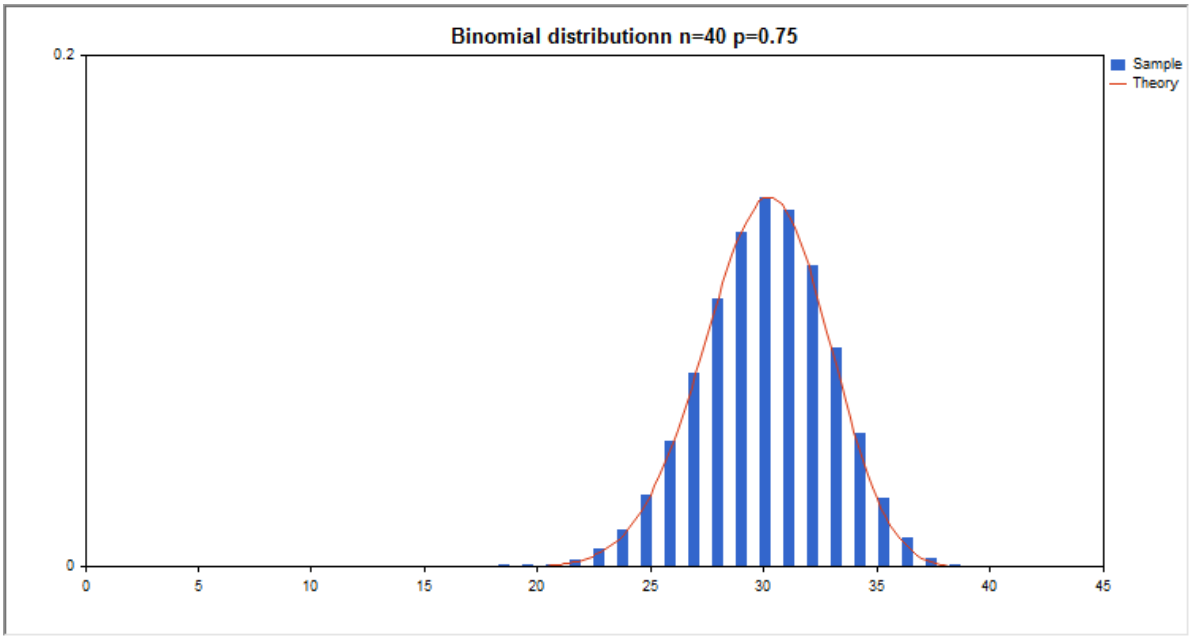
Distribuição binomial

Esta seção apresenta as funções para trabalhar com a distribuição binomial. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição binomial. A distribuição binomial é descrita pela seguinte fórmula:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

Onde:

- x — valor da variável aleatória
- n — número de testes
- p — probabilidade de sucesso para cada teste



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityBinomial	Calcula a densidade de probabilidade da distribuição binomial
MathCumulativeDistributionBinomial	Calcula o valor da função distribuição binomial de probabilidades
MathQuantileBinomial	Calcula o inverso da função de distribuição binomial para a probabilidade especificada
MathRandomBinomial	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição binomial
MathMomentsBinomial	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição binomial

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double n_par=40;        // número de tentativas
input double p_par=0.75;      // probabilidade de sucesso em cada tentativa
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=20;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição binomial
   MathRandomBinomial(n_par,p_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Binomial distributionn n=%G p=%G",n_par,p_par)
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityBinomial

Calcula a função de probabilidade (probability mass function) da distribuição binomial com parâmetros n e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valor da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucesso
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial com parâmetros n e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // valor da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucesso
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial com parâmetros n e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dbinom() no R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial com parâmetros n e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

n

[in] Parâmetro de distribuição (número de testes).

p

[in] Parâmetro de distribuição (probabilidade de sucesso para cada teste).

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

Calcula a distribuição binomial de probabilidade com parâmetros n e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,             // valor da variável aleatória
   const double  n,             // parâmetro de distribuição (número de testes)
   const double  p,             // parâmetro de distribuição (probabilidade de sucesso
   const bool    tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição binomial de probabilidade com parâmetros n e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionBinomial(
   const double  x,             // valor da variável aleatória
   const double  n,             // parâmetro de distribuição (número de testes)
   const double  p,             // parâmetro de distribuição (probabilidade de sucesso
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição binomial de probabilidade com parâmetros n e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pweibull() no R.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   const bool    tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição binomial de probabilidade com parâmetros n e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

n

[in] Parâmetro de distribuição (número de testes).

p

[in] Parâmetro de distribuição (probabilidade de sucesso para cada teste).

tail

[in] Sinalizador de cálculo. Se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileBinomial

Calcula o inverso da função de distribuição binomial com parâmetros n e p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição binomial com parâmetros n e p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileBinomial(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição binomial com parâmetros n e p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qbinom() no R.

```
double  MathQuantileBinomial(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição binomial com parâmetros n e p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileBinomial(
   const double& probability[],  // matriz com valores de probabilidade da variável al
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

n

[in] Parâmetro de distribuição (número de testes).

p

[in] Parâmetro de distribuição (probabilidade de sucesso para cada teste).

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomBinomial

Gera um número pseudo-aleatório distribuído de acordo com a distribuição binomial com parâmetros n e p. Em caso de erro, retorna NaN.

```
double  MathRandomBinomial(
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição binomial com parâmetros n e p. Em caso de erro, retorna false. Equivalente a rweibull() no R.

```
bool  MathRandomBinomial(
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

n

[in] Parâmetro de distribuição (número de testes).

p

[in] Parâmetro de distribuição (probabilidade de sucesso para cada teste).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsBinomial

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição binomial com parâmetros n e p.

```
double  MathMomentsBinomial(
   const double  n,              // parâmetro de distribuição (número de testes)
   const double  p,              // parâmetro de distribuição (probabilidade de sucess
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

n

[in] Parâmetro de distribuição (número de testes).

p

[in] Parâmetro de distribuição (probabilidade de sucesso para cada teste).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

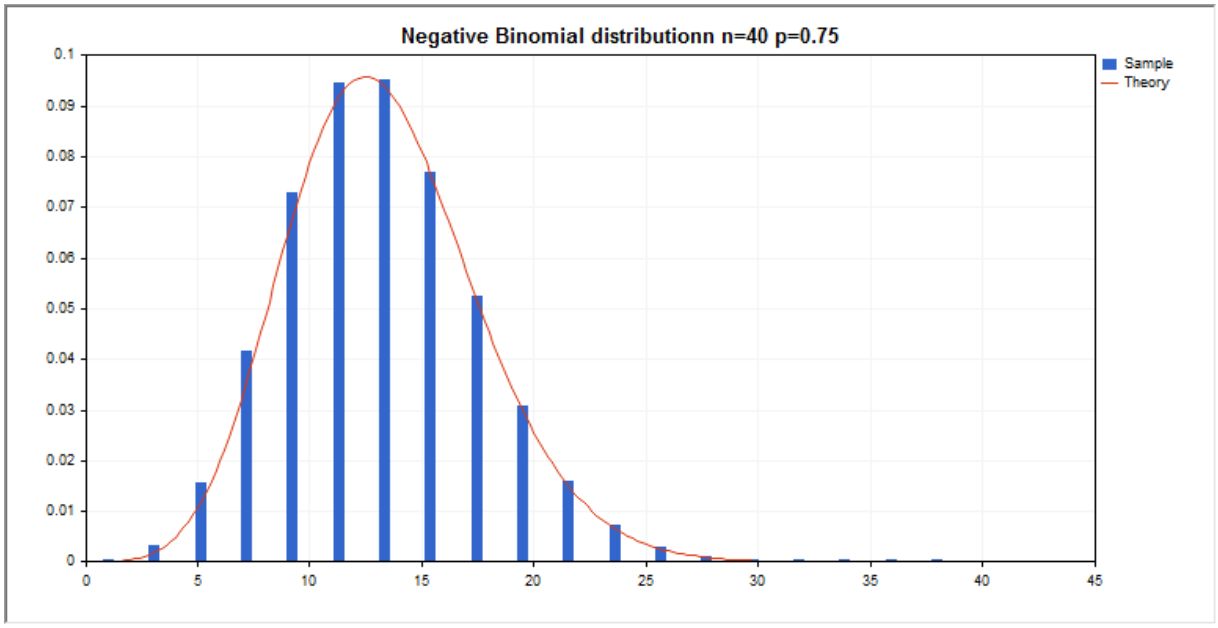
Distribuição binomial negativa

Esta seção apresenta as funções para trabalhar com a distribuição binomial negativa. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição binomial negativa. A distribuição binomial negativa é descrita pela seguinte fórmula:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

Onde:

- x — valor da variável aleatória
- r — número de testes bem-sucedidos
- p — probabilidade de sucesso



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityNegativeBinomial	Calcula a densidade de probabilidade da distribuição binomial negativa
MathCumulativeDistributionNegativeBinomial	Calcula o valor da função distribuição binomial negativa de probabilidades
MathQuantileNegativeBinomial	Calcula o inverso da função de distribuição binomial negativa para a probabilidade especificada
MathRandomNegativeBinomial	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição binomial negativa
MathMomentsNegativeBinomial	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição binomial negativa

© 2000-2024, MetaQuotes Ltd.

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double n_par=40;        // número de tentativas
input double p_par=0.75;      // probabilidade de sucesso em cada tentativa
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=19;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição binomial negativa
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Negative Binomial distributionn n=%G p=%G",n_p
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

Calcula a função de probabilidade (probability mass function) da distribuição binomial negativa com parâmetros r e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  r,             // número de testes bem-sucedidos
   const double  p,             // probabilidade de sucesso
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial negativa com parâmetros r e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  r,             // número de testes bem-sucedidos
   const double  p,             // probabilidade de sucesso
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial negativa com parâmetros r e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dnbinom() no R.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição binomial negativa com parâmetros r e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da variável aleatória.

r

[in] Número de testes bem-sucedidos

p

[in] Probabilidade de sucesso.

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

Calcula a distribuição binomial negativa de probabilidade com parâmetros r e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  r,             // número de testes bem-sucedidos
   const double  p,             // probabilidade de sucesso
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição binomial negativa de probabilidade com parâmetros r e p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  r,             // número de testes bem-sucedidos
   const double  p,             // probabilidade de sucesso
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a distribuição binomial negativa de probabilidade com parâmetros r e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pweibull() no R.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a distribuição binomial negativa de probabilidade com parâmetros r e p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // matriz com valores da variável aleatória
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

r

[in] Número de testes bem-sucedidos.

p

[in] Probabilidade de sucesso.

tail

[in] Sinalizador de cálculo, se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo de logaritmo do valor, se log_mode=true, é retornado o valor do logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileNegativeBinomial

Calcula o inverso da função de distribuição binomial negativa com parâmetros r e p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição binomial negativa com parâmetros r e p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição binomial negativa com parâmetros r e p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qnbinom() no R.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição binomial negativa com parâmetros r e p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

© 2000-2024, MetaQuotes Ltd.

[in] Matriz com valores da probabilidade da variável aleatória.

r

[in] Número de testes bem-sucedidos.

p

[in] Probabilidade de sucesso.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomNegativeBinomial

Gera um número pseudo-aleatório distribuído de acordo com a distribuição binomial negativa com parâmetros r e p. Em caso de erro, retorna NaN.

```
double  MathRandomNegativeBinomial(
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição binomial negativa com parâmetros r e p. Em caso de erro, retorna false. Equivalente a rweibull() no R.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

r

[in] Número de testes bem-sucedidos.

p

[in] Probabilidade de sucesso.

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsNegativeBinomial

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição binomial negativa com parâmetros r e p.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // número de testes bem-sucedidos
   const double  p,              // probabilidade de sucesso
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

r

[in] Número de testes bem-sucedidos.

p

[in] Probabilidade de sucesso.

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

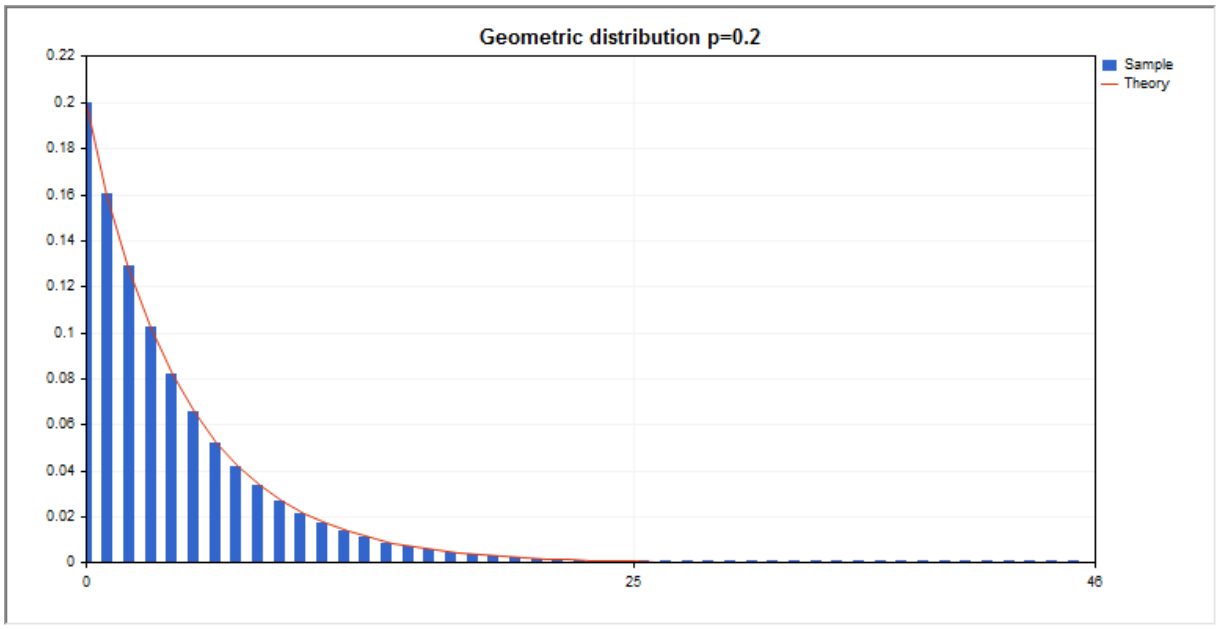
Distribuição geométrica

Esta seção apresenta as funções para trabalhar com a distribuição geométrica. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição geométrica. A distribuição geométrica é descrita pela seguinte fórmula:

$$f_{Geometric}(x \mid p) = p(1-p)^x$$

Onde:

- x — valor da variável aleatória (inteiro)
- p — probabilidade de evento em uma experiência



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityGeometric	Calcula a densidade de probabilidade da distribuição geométrica
MathCumulativeDistributionGeometric	Calcula o valor da função distribuição geométrica de probabilidades
MathQuantileGeometric	Calcula o inverso da função de distribuição geométrica para a probabilidade especificada
MathRandomGeometric	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição geométrica
MathMomentsGeometric	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição geométrica

Exemplo:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2024, MetaQuotes Ltd.

```
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double p_par=0.2;      // probabilidade de evento numa experiência
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=47;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição geométrica
   MathRandomGeometric(p_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo X
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityGeometric

Calcula a função de probabilidade (probability mass function) da distribuição geométrica com parâmetro p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valor da variável aleatória (inteiro)
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   const bool    log_mode,        // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição geométrica com parâmetro p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // valor da variável aleatória (inteiro)
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição geométrica com parâmetro p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dgeom() no R.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição geométrica com parâmetro p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

p

[in] Parâmetro de distribuição (probabilidade de evento em uma experiência).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

Calcula a função distribuição geométrica com parâmetro p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  p,             // parâmetro de distribuição (probabilidade de evento
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função distribuição geométrica com parâmetro p para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionGeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  p,             // parâmetro de distribuição (probabilidade de evento
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função distribuição geométrica com parâmetro p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a pgeom() no R.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Calcula a função distribuição geométrica com parâmetro p para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   double&       result[]        // matriz para valores da função de probabilidade
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

p

[in] Parâmetro de distribuição (probabilidade de evento em uma experiência).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se tail=true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo de logaritmo do valor, se log_mode=true, é retornado o logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para valores da função de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathQuantileGeometric

Calcula o inverso da função de distribuição geométrica com parâmetro p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  p,              // parâmetro de distribuição (probabilidade de event
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição geométrica com parâmetro p para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileGeometric(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  p,              // parâmetro de distribuição (probabilidade de event
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição geométrica com parâmetro p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qgeom() no R.

```
double  MathQuantileGeometric(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  p,              // parâmetro de distribuição (probabilidade de event
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição geométrica com parâmetro p para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileGeometric(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  p,              // parâmetro de distribuição (probabilidade de event
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

p

[in] Parâmetro de distribuição (probabilidade de evento em uma experiência).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se for false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomGeometric

Gera um número pseudo-aleatório distribuído de acordo com a distribuição geométrica com parâmetro p. Em caso de erro, retorna NaN.

```
double  MathRandomGeometric(
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição geométrica com parâmetro p. Em caso de erro, retorna false. Equivalente a rgeom() no R.

```
bool  MathRandomGeometric(
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatório
   );
```

Parâmetros

p

[in] Parâmetro de distribuição (probabilidade de evento em uma experiência).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsGeometric

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição geométrica com parâmetro p.

```
double  MathMomentsGeometric(
   const double  p,              // parâmetro de distribuição (probabilidade de evento
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

p

[in] Parâmetro de distribuição (probabilidade de evento em uma experiência).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

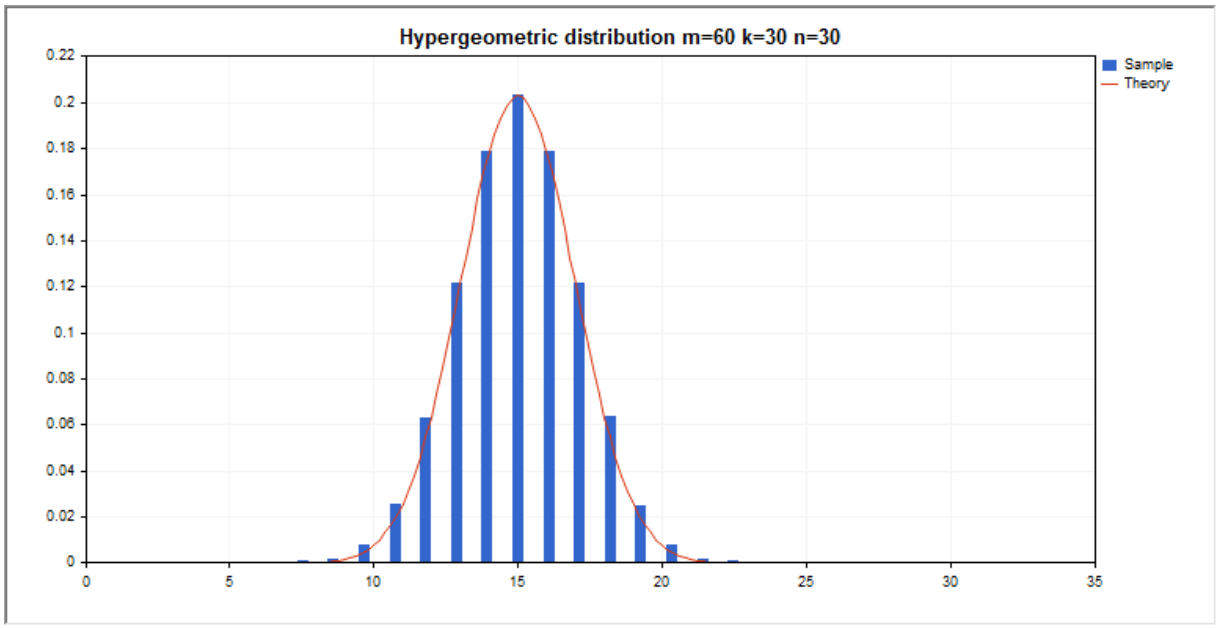
Distribuição hipergeométrica

Esta seção apresenta as funções para trabalhar com a distribuição hipergeométrica. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição hipergeométrica. A distribuição hipergeométrica é descrita pela seguinte fórmula:

$$f_{Hypergeometric}(x \mid m,k,n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

Onde:

- x — valor da variável aleatória (inteiro)
- m — número total de objetos
- k — número de objetos com a caraterística desejada
- n — número de objetos tomados



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityHypergeometric	Calcula a densidade de probabilidade da distribuição hipergeométrica
MathCumulativeDistributionHypergeometric	Calcula o valor da função distribuição hipergeométrica de probabilidades
MathQuantileHypergeometric	Calcula o inverso da função de distribuição hipergeométrica para a probabilidade especificada
MathRandomHypergeometric	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição hipergeométrica

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathMomentsHypergeometric	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição hipergeométrica

Exemplo:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double m_par=60;      // número total de objetos
input double k_par=30;      // número de objetos com a caraterística desejada
input double n_par=30;      // número de objetos tomados
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // número de valores na amostragem
   int ncells=15;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição hipergeométrica
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2024, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Hypergeometric distribution m=%G k=%G n=%G",m_
   graphic.BackgroundMainSize(16);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

Calcula a função de probabilidade (probability mass function) da distribuição hipergeométrica com parâmetros m, k e n para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (int
   const double  n,             // número de objetos tomados (inteiro)
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, i
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição hipergeométrica com parâmetros m, k e n para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (int
   const double  n,             // número de objetos tomados (inteiro)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição hipergeométrica com parâmetros m, k e n para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dhyper() no R.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (in
   const double  n,              // número de objetos tomados (inteiro)
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição hipergeométrica com parâmetros m, k e n para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // matriz com valores da variável aleatória
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (in
   const double  n,              // número de objetos tomados (inteiro)
   double&       result[]        // matriz para o valor da função densidade de probabi
   );
```

Parâmetros

© 2000-2024, MetaQuotes Ltd.

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

m

[in] Número total de objetos (inteiro).

k

[in] Número de objetos com a caraterística desejada (inteiro)

n

[in] Número de objetos tomados (inteiro)

log_mode

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

Calcula a função distribuição hipergeométrica com parâmetros m, k e n para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (int
   const double  n,             // número de objetos tomados (inteiro)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função distribuição hipergeométrica com parâmetros m, k e n para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (int
   const double  n,             // número de objetos tomados (inteiro)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função distribuição hipergeométrica com parâmetros m, k e n para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dhyper() no R.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],           // matriz com valores da variável aleatória
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (in
   const double  n,             // número de objetos tomados (inteiro)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]       // matriz para os valores da função de distribuição
   );
```

Calcula a função distribuição hipergeométrica com parâmetros m, k e n para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],           // matriz com valores da variável aleatória
   const double  m,             // número total de objetos (inteiro)
   const double  k,             // número de objetos com a caraterística desejada (in
   const double  n,             // número de objetos tomados (inteiro)
   double&       result[]       // matriz para os valores da função de distribuição
   );
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

m

[in] Número total de objetos (inteiro).

k

[in] Número de objetos com a caraterística desejada (inteiro)

n

[in] Número de objetos tomados (inteiro)

tail

[in] Sinalizador de cálculo, se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo de logaritmo do valor, se log_mode=true, é retornado o valor do logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função de distribuição.

© 2000-2024, MetaQuotes Ltd.

MathQuantileHypergeometric

Calcula o inverso da função de distribuição hipergeométrica com parâmetros m, k e n para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição hipergeométrica com parâmetros m, k e n para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantileHypergeometric(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição hipergeométrica com parâmetros m, k e n para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qhyper() no R.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo s
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcu
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição hipergeométrica com parâmetros m, k e n para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   double&       result[]        // matriz com valores de quantis
   );
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

m

[in] Número total de objetos (inteiro).

k

[in] Número de objetos com a caraterística desejada (inteiro)

n

[in] Número de objetos tomados (inteiro)

tail

[in] Sinalizador de cálculo, se tail=false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomHypergeometric

Gera um número pseudo-aleatório distribuído de acordo com a distribuição hipergeométrica com parâmetros m, n e k. Em caso de erro, retorna NaN.

```
double  MathRandomHypergeometric(
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   int&          error_code      // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição hipergeométrica com parâmetros m, n e k. Em caso de erro, retorna false. Equivalente a rgeom() no R.

```
bool  MathRandomHypergeometric(
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   const int     data_count,     // número de dados necessários
   double&       result[]        // matriz com os valores dos números pseudo-aleatóri
   );
```

Parâmetros

m

[in] Número total de objetos (inteiro).

k

[in] Número de objetos com a caraterística desejada (inteiro)

n

[in] Número de objetos tomados (inteiro)

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsHypergeometric

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição hipergeométrica com parâmetros m, n e k.

```
double  MathMomentsHypergeometric(
   const double  m,              // número total de objetos (inteiro)
   const double  k,              // número de objetos com a caraterística desejada (i
   const double  n,              // número de objetos tomados (inteiro)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

m

[in] Número total de objetos (inteiro).

k

[in] Número de objetos com a caraterística desejada (inteiro)

n

[in] Número de objetos tomados (inteiro)

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

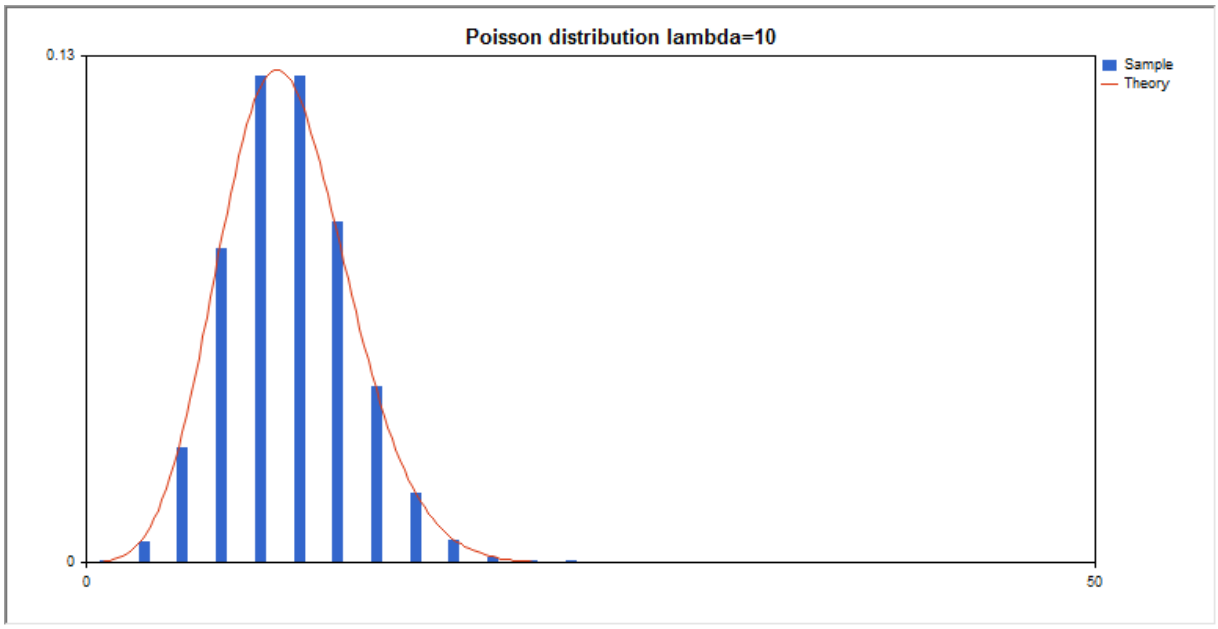
Distribuição de Poisson

Esta seção contém funções para trabalhar com a distribuição de Poisson. Elas permitem que você calcule a densidade, probabilidade, quantis e gere números pseudo-aleatórios de acordo com a distribuição de Poisson. A distribuição de Poisson é descrita pela seguinte fórmula:

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

Onde:

- x — valor da variável aleatória
- λ — parâmetro de distribuição (mean)



Além do cálculo das variáveis aleatórias individuais, é possível trabalhar com suas matrizes.

Função	Descrição
MathProbabilityDensityPoisson	Calcula a densidade de probabilidade da distribuição de Poisson
MathCumulativeDistributionPoisson	Calcula o valor da função distribuição de Poisson de probabilidades
MathQuantilePoisson	Calcula o inverso da função de distribuição de Poisson para a probabilidade especificada
MathRandomPoisson	Gera um valor/matriz pseudo-aleatório de valores pseudo-aleatórios de acordo com a distribuição de Poisson
MathMomentsPoisson	Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Poisson

Exemplo:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2024, MetaQuotes Ltd.

```
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- input parameters
input double lambda_par=10;      // parâmetro de distribuição (mean)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- desativamos a exibição do gráfico de preço
   ChartSetInteger(0,CHART_SHOW,false);
//--- inicializamos o gerador de números aleatórios
   MathSrand(GetTickCount());
//--- geramos a amostragem da variável aleatória
   long chart=0;
   string name="GraphicNormal";
   int n=100000;       // número de valores na amostragem
   int ncells=13;       // número de intervalos no gráfico de colunas
   double x[];          // centros de intervalos no gráfico de colunas
   double y[];          // número de valores a partir da amostragem dentro do interval
   double data[];       // amostragem de valores aleatórios
   double max,min;      // valores máximo e mínimo na amostragem
//--- obtemos a amostragem a partir da distribuição de Poisson
   MathRandomPoisson(lambda_par,n,data);
//--- calculamos os dados para plotagem do gráfico de colunas
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- obtemos os limites de sequência e passo para plotagem da curva teórica
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- obtemos teoricamente os cálculos no intervalo [min,max]
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- dimensionamos
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- imprimimos o gráfico
   CGraphic graphic;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%G",lambda_par));
```

© 2000-2024, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- desativamos o dimensionamento automático no eixo Y
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- plot all curves
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- e agora traçamos a curva teórica de densidade da distribuição
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- plot all curves
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Calculate frequencies for data set                              |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- definimos os centros dos intervalos
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- preenchemos a frequência de aparecimento no intervalo
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Calculates values for sequence generation                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- calculamos a magnitude da sequência para obter a precisão da normalização
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- normalizamos os valores máx. e mín. com a precisão
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- definimos o passo de geração da mesma maneira a partir da precisão definida
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityPoisson

Calcula a função de probabilidade (probability mass function) da distribuição de Poisson com parâmetro lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  lambda,        // parâmetro de distribuição (mean)
   const bool    log_mode,       // cálculo de logaritmo do valor, se log_mode=true, 
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição de Poisson com parâmetro lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathProbabilityDensityPoisson(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  lambda,        // parâmetro de distribuição (mean)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição de Poisson com parâmetro lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dhyper() no R.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // matriz com valores da variável aleatória
   const double  lambda,         // parâmetro de distribuição (mean)
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se 
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Calcula a função de probabilidade (probability mass function) da distribuição de Poisson com parâmetro lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // matriz com valores da variável aleatória
   const double  lambda,         // parâmetro de distribuição (mean)
   double&       result[]        // matriz para o valor da função densidade de probab
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

lambda

[in] Parâmetro de distribuição (mean).

log_mode

© 2000-2024, MetaQuotes Ltd.

[in] Sinalizador de cálculo do algoritmo de um valor. Se log_mode=true, retorna o logaritmo natural da função densidade de probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função densidade de probabilidade.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

Calcula o valor da função de distribuição de Poisson com parâmetro lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  lambda,        // parâmetro de distribuição (mean)
   const double  tail,          // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,      // sinalizador de cálculo de logaritmo do valor, se lo
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula o valor da função de distribuição de Poisson com parâmetro lambda para a variável aleatória x. Em caso de erro, retorna NaN.

```
double  MathCumulativeDistributionPoisson(
   const double  x,             // valor da variável aleatória (inteiro)
   const double  lambda,        // parâmetro de distribuição (mean)
   int&          error_code     // variável para registro do código de erro
   );
```

Calcula o valor da função de distribuição de Poisson com parâmetro lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false. Equivalente a dhyper() no R.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // matriz com valores da variável aleatória
   const double  lambda,         // parâmetro de distribuição (mean)
   const double  tail,           // sinalizador de cálculo, se for true, é calculada a
   const bool    log_mode,       // sinalizador de cálculo de logaritmo do valor, se l
   double&       result[]        // matriz para os valores da função de distribuição
   );
```

Calcula o valor da função de distribuição de Poisson com parâmetro lambda para a matriz das variáveis aleatórias x[]. Em caso de erro, retorna false.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // matriz com valores da variável aleatória
   const double  lambda,         // parâmetro de distribuição (mean)
   double&       result[]        // matriz para os valores da função de distribuição
   );
```

Parâmetros

x

[in] Valor da variável aleatória.

x[]

[in] Matriz com valores da variável aleatória.

lambda

[in] Parâmetro de distribuição (mean).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se for true, é calculada a probabilidade de que a variável aleatória não será superior a x.

log_mode

[in] Sinalizador de cálculo de logaritmo do valor, se log_mode=true, é retornado o valor do logaritmo natural da probabilidade.

error_code

[out] Variável para registro do código de erro.

result[]

[out] Matriz para os valores da função de distribuição.

© 2000-2024, MetaQuotes Ltd.

MathQuantilePoisson

Calcula o inverso da função de distribuição de Poisson com parâmetro lambda para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  lambda,         // parâmetro de distribuição (mean)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Poisson com parâmetro lambda para a probabilidade *probability*. Em caso de erro, retorna NaN.

```
double  MathQuantilePoisson(
   const double  probability,    // valor da probabilidade de aparecimento de uma var
   const double  lambda,         // parâmetro de distribuição (mean)
   int&          error_code      // variável para registro do código de erro
   );
```

Calcula o inverso da função de distribuição de Poisson com parâmetro lambda para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false. Equivalente a qhyper() no R.

```
double  MathQuantilePoisson(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  lambda,         // parâmetro de distribuição (mean)
   const bool    tail,           // sinalizador de cálculo, se for false, o cálculo se
   const bool    log_mode,       // sinalizador de cálculo, se log_mode=true, o cálcul
   double&       result[]        // matriz com valores de quantis
   );
```

Calcula o inverso da função de distribuição de Poisson com parâmetro lambda para a matriz de valores da probabilidade *probability[]*. Em caso de erro, retorna false.

```
bool  MathQuantilePoisson(
   const double& probability[],  // matriz com valores de probabilidade da variável a
   const double  lambda,         // parâmetro de distribuição (mean)
   double&       result[]        // matriz com valores de quantis
   );
```

Parâmetros

probability

[in] Valor da probabilidade da variável aleatória.

probability[]

[in] Matriz com valores da probabilidade da variável aleatória.

lambda

[in] Parâmetro de distribuição (mean).

© 2000-2024, MetaQuotes Ltd.

tail

[in] Sinalizador de cálculo, se tail=false, o cálculo é realizado para a probabilidade 1.0 - probability.

log_mode

[in] Sinalizador de cálculo, se log_mode=true, o cálculo é realizado para a probabilidade Exp(probability).

error_code

[out] Variável para obter o código de erro.

result[]

[out] Matriz com os valores dos quantis.

© 2000-2024, MetaQuotes Ltd.

MathRandomPoisson

Gera um número pseudo-aleatório distribuído de acordo com a distribuição de Poisson com parâmetro lambda. Em caso de erro, retorna NaN.

```
double  MathRandomPoisson(
   const double  lambda,          // parâmetro de distribuição (mean)
   int&          error_code       // variável para registro do código de erro
   );
```

Gera números pseudo-aleatórios distribuídos de acordo com a distribuição de Poisson com parâmetro lambda. Em caso de erro, retorna false. Equivalente a rgeom() no R.

```
bool  MathRandomPoisson(
   const double  lambda,          // parâmetro de distribuição (mean)
   const int     data_count,      // número de dados necessários
   double&       result[]         // matriz com os valores dos números pseudo-aleatóric
   );
```

Parâmetros

lambda

[in] Parâmetro de distribuição (mean).

error_code

[out] Variável para registro do código de erro.

data_count

[out] Número de dados necessários.

result[]

[out] Matriz para obter os valores dos números pseudo-aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathMomentsPoisson

Calcula os valores numéricos teóricos dos primeiros 4 momentos da distribuição de Poisson com parâmetro lambda.

```
double  MathMomentsPoisson(
   const double  lambda,         // parâmetro de distribuição (mean)
   double&       mean,           // variável para o valor médio
   double&       variance,       // variável para a dispersão
   double&       skewness,       // variável para a obliquidade
   double&       kurtosis,       // variável para a curtose
   int&          error_code      // variável para o código de erro
   );
```

Parâmetros

lambda

[in] Parâmetro de distribuição (mean).

mean

[out] Variável para obter o valor médio.

variance

[out] Variável para obter a dispersão.

skewness

[out] Variável para obter a obliquidade.

kurtosis

[out] Variável para obter a curtose.

error_code

[out] Variável para obter o código de erro.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

Funções auxiliares

Grupo de funções que executam operações matemáticas básicas: cálculo da função gama, função beta, fatoriais, exponenciais, logaritmos com diferentes bases, raiz quadrada, etc.

Fornecidas possibilidades de processamento como valores numéricos separados (reais e inteiros), bem como suas matizes (com resultados registrados na matriz original ou numa separada).

Função	Descrição
MathRandomNonZero	Retorna um número aleatório entre 0,0 a 1,0 com vírgula flutuante.
MathMoments	Calcula os primeiros 4 momentos dos elementos da matriz: média, dispersão, obliquidade, curtose.
MathPowInt	Gera um número na potência inteira especificada.
MathFactorial	Calcula o fatorial do número inteiro especificado.
MathTrunc	Calcula a parte integrante do número especificado ou os elementos da matriz.
MathRound	Arredonda um número ou uma matriz de números para um número especificado de casas decimais.
MathArctan2	Calcula o ângulo cuja tangente é a relação entre dois números no intervalo [-pi, pi].
MathGamma	Calcula o valor da função gama.
MathGammaLog	Calcula o logaritmo da função gama.
MathBeta	Calcula o valor da função beta.
MathBetaLog	Calcula o valor do logaritmo da função beta.
MathBetaIncomplete	Ele calcula o valor da função beta incompleta.
MathGammaIncomplete	Ele calcula o valor da função gama incompleta.
MathBinomialCoefficient	Calcula o coeficiente binomial.
MathBinomialCoefficientLog	Calcula o logaritmo do coeficiente binomial.
MathHypergeometric2F2	Calcula o valor da função hipergeométrica.
MathSequence	Gera uma sequência de valores com base em: o primeiro elemento, o último elemento, o passo da sequência.

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathSequenceByCount	Gera uma sequência de valores com base em: o primeiro elemento, o segundo elemento, o número de elementos na sequência.
MathReplicate	Gera uma sequência repetitiva de valores.
MathReverse	Gera uma matriz de valores com os elementos na ordem inversa.
MathIdentical	Compara dois matrizes de valores e retorna true, se todos os elementos são os mesmos.
MathUnique	Gera uma matriz apenas com valores distintos.
MathQuickSortAscending	Função para classificação crescente.
MathQuickSortDescending	Função para classificação decrescente.
MathQuickSort	Função de classificação.
MathOrder	Cria uma matiz com um rearranjo na ordem dos elementos da matriz após a classificação.
MathBitwiseNot	Calcula o resultado da operação binária NOT para elementos de matriz.
MathBitwiseAnd	Calcula o resultado da operação binária AND para elementos de matriz.
MathBitwiseOr	Calcula o resultado da operação binária OR para elementos de matriz.
MathBitwiseXor	Calcula o resultado da operação binária XOR para elementos de matriz.
MathBitwiseShiftL	Calcula o resultado da operação binária SHL para elementos de matriz.
MathBitwiseShiftR	Calcula o resultado da operação binária SHR para elementos de matriz.
MathCumulativeSum	Gera uma matriz com o valor acumulado.
MathCumulativeProduct	Gera uma matriz com o produto acumulado.
MathCumulativeMin	Gera uma matriz de valores mínimos acumulados.
MathCumulativeMax	Gera uma matriz de valores máximos acumulados.
MathSin	Calcula o valor da função sin(x) para os elementos da matriz.

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathCos	Calcula o valor da função cos(x) para os elementos da matriz.
MathTan	Calcula o valor da função tan(x) para os elementos da matriz.
MathArcsin	Calcula o valor da função arcsin(x) para os elementos da matriz.
MathArccos	Calcula o valor da função arccos(x) para os elementos da matriz.
MathArctan	Calcula o valor da função arctan(x) para os elementos da matriz.
MathSinPi	Calcula o valor da função sin(pi*x) para os elementos da matriz.
MathCosPi	Calcula o valor da função cos(pi*x) para os elementos da matriz.
MathTanPi	Calcula o valor da função tan(pi*x) para os elementos da matriz.
MathAbs	Calcula o valor absoluto dos elementos da matriz.
MathCeil	Retorna o valor de número inteiro mais próximo - desde cima - para os elementos da matriz.
MathFloor	Retorna o valor de número inteiro mais próximo - desde baixo - para os elementos da matriz.
MathSqrt	Calcula a raiz quadrada para os elementos da matriz.
MathExp	Calcula o valor da função exp(x) para os elementos da matriz.
MathPow	Calcula o valor da função pow(x, power) para os elementos da matriz.
MathLog	Calcula o valor da função log(x) para os elementos da matriz.
MathLog2	Calcula o valor do logaritmo na base 2 para elementos da matriz.
MathLog10	Calcula o valor do logaritmo na base 10 para elementos da matriz.
MathDifference	Gera uma matriz com diferenças de elementos y[i]=x[i+lag]-x[i].

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathSample	Gera uma amostragem aleatória de elementos da matriz.
MathTukeySummary	Calcula o resumo de cinco números de Tukey para os elementos da matriz.
MathRange	Calcula os valores mínimos e máximos dos elementos da matriz.
MathMin	Retorna o valor mínimo de todos os elementos na matriz.
MathMax	Retorna o valor máximo de todos os elementos da matriz.
MathSum	Retorna a soma dos elementos da matriz.
MathProduct	Retorna o produto dos elementos da matriz.
MathStandardDeviation	Calcula o desvio padrão dos elementos da matriz.
MathAverageDeviation	Calcula o desvio médio dos elementos da matriz.
MathMedian	Calcula o valor de mediana dos elementos da matriz.
MathMean	Calcula o valor médio dos elementos da matriz.
MathVariance	Calcula a dispersão dos elementos da matriz.
MathSkewness	Calcula a obliquidade dos elementos da matriz.
MathKurtosis	Calcula a curtose dos elementos da matriz.
MathLog1p	Calcula o valor da função de log(1+x) para os elementos da matriz.
MathExpm1	Calcula o valor da função exp(x)-1 para os elementos da matriz.
MathSinh	Calcula o valor da função sinh(x) para os elementos da matriz.
MathCosh	Calcula o valor da função cosh(x) para os elementos da matriz.
MathTanh	Calcula o valor da função tanh(x) para os elementos da matriz.
MathArcsinh	Calcula o valor da função arcsinh(x) para os elementos da matriz.

© 2000-2024, MetaQuotes Ltd.

Função	Descrição
MathArccosh	Calcula o valor da função arccosh(x) para os elementos da matriz.
MathArctanh	Calcula o valor da função arctanh(x) para os elementos da matriz.
MathSignif	Arredonda um valor para o número especificado de dígitos na mantissa.
MathRank	Calcula a posição dos elementos da matriz.
MathCorrelationPearson	Calcula o coeficiente de correlação de Pearson.
MathCorrelationSpearman	Calcula o coeficiente de correlação de Spearman.
MathCorrelationKendall	Calcula o coeficiente de correlação de Kendall.
MathQuantile	Calcula o quantis seletivos que corresponde às probabilidades definidas.
MathProbabilityDensityEmpirical	Calcula a função densidade de probabilidade empírica para os valores aleatórios.
MathCumulativeDistributionEmpirical	Calcula a distribuição cumulativa empírica para os valores aleatórios.

© 2000-2024, MetaQuotes Ltd.

MathRandomNonZero

Retorna um número aleatório entre 0,0 a 1,0 com vírgula flutuante.

```
double  MathRandomNonZero()
```

Valor de retorno

Número aleatório entre 0,0 a 1,0 com vírgula flutuante.

© 2000-2024, MetaQuotes Ltd.

MathMoments

Calcula os primeiros 4 momentos dos elementos da matriz: média, dispersão, obliquidade, curtose.

```
bool  MathMoments(
   const double&  array[],            // matriz de valores
   double&        mean,               // variável para média
   double&        variance,           // variável para dispersão
   double&        skewness,           // variável para obliquidade
   double&        kurtosis,           // variável para curtose
   const int      start=0,            // índice inicial
   const int      count=WHOLE_ARRAY   // número de elementos
   )
```

Parâmetros

array[]

[in] Matriz de valores.

mean

[out] Variável para a média (1 momento).

variance

[out] Variável para dispersão (2 momento).

skewness

[out] Variável para a obliquidade (3 momento).

kurtosis

[out] Variável para a curtose (4 momento).

start=0

[in] Índice inicial para o cálculo.

count=WHOLE_ARRAY

[in] Número de elementos para o cálculo.

Valor de retorno

Retorna true, se os momentos forem calculados com sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

MathPowInt

Gera um número na potência inteira especificada.

```
double  MathPowInt(
   const double  x,       // valor do número
   const int     power    // exponenciação
   )
```

Parâmetros

x

[in] Número de precisão dupla com ponto flutuante e raiz quadrada.

power

[in] Inteiro que especifica a exponenciação.

Valor de retorno

Número *x* elevado à potência especificada.

© 2000-2024, MetaQuotes Ltd.

MathFactorial

Calcula o fatorial do número inteiro especificado.

```
double  MathFactorial(
   const int  n   // valor de um número
   )
```

Parâmetros

n

[in] Número inteiro cujo fatorial é necessário para calcular.

Valor de retorno

Fatorial de um número.

© 2000-2024, MetaQuotes Ltd.

MathTrunc

Calcula a parte integrante do número especificado ou os elementos da matriz.

Versão para trabalhar com um número de precisão dupla e ponto flutuante:

```
double  MathTrunc(
   const double   x          // valor do número
   )
```

Valor de retorno

Parte inteira do número definido.

Versão para trabalhar com uma matriz de números de precisão dupla e ponto flutuante. Os resultados são registrados numa nova matriz:

```
bool  MathTrunc(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Versão para trabalhar com uma matriz de números de precisão dupla e ponto flutuante. Os resultados são registrados na mesma matriz:

```
bool  MathTrunc(
   double&        array[]    // matriz de valores
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Parâmetros

x

[in] Número de precisão dupla com ponto flutuante cuja parte inteira é necessário obter.

array[]

[in] Matriz de números de precisão dupla com ponto flutuante cuja parte inteira é necessário obter.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

© 2000-2024, MetaQuotes Ltd.

MathRound

Arredonda um número de precisão dupla com ponto flutuante ou uma matriz de números para um número especificado de casas decimais.

Versão para arredondamento de um número de precisão dupla com ponto flutuante para um número especificado de casas decimais.

```
double  MathRound(
   const double   x,          // valor do número
   const int      digits      // número de dígitos após o ponto decimal
   )
```

Valor de retorno

Número mais próximo do parâmetro x, número de cifras da parte decimal que é igual a digits.

Versão para arredondamento da matriz de números de precisão dupla com ponto flutuante para um número especificado de casas decimais. Os resultados são registrados numa nova matriz.

```
bool  MathRound(
   const double&  array[],    // matriz de valores
   int            digits,     // número de dígitos após o ponto decimal
   double&        result[]    // matriz de resultado
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Versão para arredondamento da matriz de números de precisão dupla com ponto flutuante para um número especificado de casas decimais. Os resultados são registrados na mesma matriz.

```
bool  MathRound(
   double&        array[],    // matriz de valores
   int            digits      // número de dígitos após o ponto decimal
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Parâmetros

x

[in] Número - a ser arredondado - de precisão dupla com ponto flutuante.

digits

[in] Número de dígitos da parte decimal no valor a ser retornado.

array[]

[in] Matriz de números - a ser arredondados - de precisão dupla com ponto flutuante.

array[]

© 2000-2024, MetaQuotes Ltd.

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

© 2000-2024, MetaQuotes Ltd.

MathArctan2

Retorna o arco tangente do quociente de dois argumentos (x, y).

Versão para trabalhar com a relação entre dois números especificados (x, y):

```
double  MathArctan2(
   const double       y,          // coordenada y
   const double       x           // coordenada x
   )
```

Valor de retorno

Ângulo θ, medido em radianos, de tal modo que -π≤θ≤π, e tan (θ) = y ou x, onde (x, y) é um ponto no sistema de coordenadas cartesianas.

Versão para trabalhar com a relação entre um par de elementos das matrizes x e y:

```
bool  MathArctan2(
   const double&      x[],        // matriz de valores x
   const double&      y[],        // matriz de valores y
   double&            result[]    // matriz de resultado
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Parâmetros

y

[in] Coordenada y do ponto.

x

[in] Coordenada x do ponto.

x[]

[in] Matriz das coordenadas x dos pontos.

y[]

[in] Matriz de coordenadas y dos pontos.

result[]

[out] Matriz para registro de resultados

Comentário:

Observe o seguinte.

- Para (x, y) no quadrante 1 o valor retornado será: 0 < θ < π/2.
- Para (x, y) no quadrante 2 o valor retornado será: π/2 < θ≤π.
- Para (x, y) no quadrante 3 o valor retornado será: -π < θ <-π/2.
- Para (x, y) no quadrante 4 o valor retornado será: -π/2 < θ < 0.

© 2000-2024, MetaQuotes Ltd.

Para os pontos fora dos quadrantes indicados, o valor retornado é especificado abaixo.

- Se y é igual a 0 e x não é negativo, θ = 0.
- Se y é igual a 0 e x é negativo, θ = π.
- Se y é um número positivo, e x é igual a 0, θ = π/2.
- Se y é negativo e x é igual a 0, θ = -π/2.
- Se y é igual a 0 e x é igual a 0, θ = -π/2.

Quer se o valor do parâmetro x ou y é igual a NaN, quer se os valores dos parâmetros x e y são iguais ao valor PositiveInfinity ou NegativeInfinity, o método retorna o valor NaN.

© 2000-2024, MetaQuotes Ltd.

MathGamma

Calcula o valor da função gama para o argumento real *x*.

```
double  MathGamma(
   const double  x       // argumento da função
   )
```

Parâmetros

x

[in] Argumento real da função.

Valor de retorno

Valor de uma função gama.

© 2000-2024, MetaQuotes Ltd.

MathGammaLog

Calcula o valor da função gama para o argumento real x.

```
double  MathGammaLog(
   const double  x      // argumento da função
   )
```

Parâmetros

x

[in] Argumento real da função.

Valor de retorno

O valor do logaritmo da função.

© 2000-2024, MetaQuotes Ltd.

MathBeta

Calcula o valor da função beta para os argumentos reais a e b.

```
double  MathBeta(
   const double  a,       // primeiro argumento da função
   const double  b        // segundo argumento da função
   )
```

Parâmetros

a

[in] Argumento da função a.

b

[in] Argumento da função b.

Valor de retorno

Valor da função.

© 2000-2024, MetaQuotes Ltd.

MathBetaLog

Calcula o valor do logaritmo da função beta para os argumentos reais a e b.

```
double  MathBetaLog(
   const double  a,       // primeiro argumento da função
   const double  b        // segundo argumento da função
   )
```

Parâmetros

a

[in] Argumento da função a.

b

[in] Argumento da função b.

Valor de retorno

O valor do logaritmo da função.

© 2000-2024, MetaQuotes Ltd.

MathBetaIncomplete

Ele calcula o valor da função beta incompleta.

```
double  MathBetaIncomplete(
   const double  x,       // argumento da função
   const double  p,       // primeiro parâmetro da função
   const double  q        // segundo parâmetro da função
   )
```

Parâmetros

x

[in] Argumento da função.

p

[in] O primeiro parâmetro da função beta deve ser >0.0.

q

[in] O segundo parâmetro da função beta deve ser >0.0.

Valor de retorno

Valor da função.

© 2000-2024, MetaQuotes Ltd.

MathGammaIncomplete

Calcula o valor da função gama incompleta.

```
double  MathGammaIncomplete(
   double  x,          // argumento da função
   double  alpha       // parâmetro da função
   )
```

Parâmetros

x

[in] Argumento da função.

alpha

[in] Parâmetro da função gama incompleta.

Valor de retorno

Valor da função.

© 2000-2024, MetaQuotes Ltd.

MathBinomialCoefficient

Calcula o coeficiente binomial: C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,       // número total de elementos
   const int  k        // número de elementos combinados
   )
```

Parâmetros

n

[in] Número de elementos.

k

[in] Número de elementos para cada combinação.

Valor de retorno

Número de combinações a partir de N de acordo com K.

© 2000-2024, MetaQuotes Ltd.

MathBinomialCoefficientLog

Calcula o logaritmo do coeficiente binomial: Log(C(n,k))=Log(n!/(k!*(n-k)!))

Versão para argumentos inteiros:

```
double  MathBinomialCoefficientLog(
   const int     n,       // número total de elementos
   const int     k        // número de elementos combinados
   )
```

Versão para argumentos reais:

```
double  MathBinomialCoefficientLog(
   const double  n,       // número total de elementos
   const double  k        // número de elementos combinados
   )
```

Parâmetros

n

[in] Número de elementos.

k

[in] Número de elementos para cada combinação.

Valor de retorno

Logaritmo de C(n,k).

© 2000-2024, MetaQuotes Ltd.

MathHypergeometric2F2

Calcula o valor da função Hypergeometric_2F2 (a, b, c, d, z), usando o método de Taylor.

```
double  MathHypergeometric2F2(
   const double  a,      // primeiro parâmetro da função
   const double  b,      // segundo parâmetro da função
   const double  c,      // terceiro parâmetro da função
   const double  d,      // quarto parâmetro da função
   const double  z       // quinto parâmetro da função
   )
```

Parâmetros

a

[in] Primeiro parâmetro da função.

b

[in] Segundo parâmetro da função.

c

[in] Terceiro parâmetro da função.

d

[in] Quarto parâmetro da função.

z

[in] Quinto parâmetro da função.

Valor de retorno

Valor da função.

© 2000-2024, MetaQuotes Ltd.

MathSequence

Gera uma sequência de valores com base em: o primeiro elemento, o último elemento, o passo da sequência.

Versão para trabalhar com valores reais:

```
bool  MathSequence(
   const double  from,        // valor inicial
   const double  to,          // valor final
   const double  step,        // passo
   double&       result[]     // matriz de resultado
   )
```

Versão para trabalhar com valores inteiros:

```
bool  MathSequence(
   const int     from,        // valor inicial
   const int     to,          // valor final
   const int     step,        // passo
   int&          result[]     // matriz de resultado
   )
```

Parâmetros

from

[in] Primeiro valor da sequência

to

[in] Último valor da sequência

step

[in] Passo da sequência.

result[]

[out] Matriz para registro de sequência.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSequenceByCount

Gera uma sequência de valores com base em: o primeiro elemento, o último elemento, o passo da sequência.

Versão para trabalhar com valores reais:

```
bool  MathSequenceByCount(
   const double  from,        // valor inicial
   const double  to,          // valor final
   const int     count,       // quantidade
   double&       result[]     // matriz de resultado
   )
```

Versão para trabalhar com valores inteiros:

```
bool  MathSequenceByCount(
   const int     from,        // valor inicial
   const int     to,          // valor final
   const int     count,       // quantidade
   int&          result[]     // matriz de resultado
   )
```

Parâmetros

from

[in] Primeiro valor da sequência.

to

[in] Último valor da sequência.

count

[in] Número de elementos da sequência.

result[]

[out] Matriz para registro de sequência.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathReplicate

Gera uma sequência repetitiva de valores.

Versão para trabalhar com valores reais:

```
bool  MathReplicate(
   const double&  array[],    // matriz de valores
   const int      count,      // número de vezes
   double&        result[]    // matriz de resultado
   )
```

Versão para trabalhar com valores inteiros:

```
bool  MathReplicate(
   const int&     array[],    // matriz de valores
   const int      count,      // número de vezes
   int&           result[]    // matriz de resultado
   )
```

Parâmetros

array[]

[in] Matriz para geração de sequência.

count

[in] Número de repetições de matriz na sequência.

result[]

[out] Matriz para registro de sequência.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathReverse

Gera uma matriz de valores com os elementos na ordem inversa.

Versão para trabalhar com valores reais e armazenamento de resultados numa nova matriz:

```
bool  MathReverse(
   const double&  array[],      // matriz de valores
   double&        result[]      // matriz de resultado
   )
```

Versão para trabalhar com valores inteiros e armazenamento de resultados numa nova matriz:

```
bool  MathReverse(
   const int&     array[],      // matriz de valores
   int&           result[]      // matriz de resultado
   )
```

Versão para trabalhar com valores reais e armazenamento de resultados na mesma matriz.

```
bool  MathReverse(
   double&        array[]       // matriz de valores
   )
```

Versão para trabalhar com valores inteiros e armazenamento de resultados na mesma matriz.

```
bool  MathReverse(
   int&           array[]       // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de saída com valores na ordem inversa.

result[]

[out] Matriz de saída com valores na ordem inversa.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathIdentical

Compara dois matrizes de valores e retorna true, se todos os elementos são os mesmos.

Versão para trabalhar com matrizes de valores reais:

```
bool  MathIdentical(
   const double&  array1[],      // primeira matriz de valores
   const double&  array2[]       // segunda matriz de valores
   )
```

Versão para trabalhar com matrizes de valores inteiros:

```
bool  MathIdentical(
   const int&     array1[],      // primeira matriz de valores
   const int&     array2[]       // segunda matriz de valores
   )
```

Parâmetros

array1[]

[in] Primeira matriz para comparação.

array2[]

[in] Segunda matriz para comparação.

Valor de retorno

Retorna true, se as matrizes são iguais, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathUnique

Gera uma matriz apenas com valores distintos.

Versão para trabalhar com valores reais:

```
bool  MathUnique(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultado
   )
```

Versão para trabalhar com valores inteiros:

```
bool  MathUnique(
   const int&     array[],      // matriz de valores
   int&           result[]      // matriz de resultado
   )
```

Parâmetros

array[]

[in] Matriz de origem.

result[]

[out] Matriz para registro de valores exclusivos.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathQuickSortAscending

Função para classificação ascendente simultânea de matrizes array[] e indices[], usando o algoritmo QuickSort.

```
void  MathQuickSortAscending(
   double&  array[],      // matriz de valores
   int&     indices[],    // matriz de índices
   int      first,        // valor inicial
   int      last          // valor final
   )
```

Parâmetros

array[]

[in][out] Matriz para classificação.

indices[]

[in][out] Matriz para armazenar índices da matriz de origem.

first

[in] Índice do elemento a partir do qual é necessário começar a classificação.

last

[in] Índice do elemento no qual é necessário acabar a classificação.

© 2000-2024, MetaQuotes Ltd.

MathQuickSortDescending

Função para classificação descendente simultânea de matrizes array[] e indices[], usando o algoritmo QuickSort.

```
void  MathQuickSortDescending(
   double&  array[],      // matriz de valores
   int&     indices[],    // matriz de índices
   int      first,        // valor inicial
   int      last          // valor final
   )
```

Parâmetros

array[]

[in][out] Matriz para classificação.

indices[]

[in][out] Matriz para armazenar índices da matriz de origem.

first

[in] Índice do elemento a partir do qual é necessário começar a classificação.

last

[in] Índice do elemento no qual é necessário acabar a classificação.

© 2000-2024, MetaQuotes Ltd.

MathQuickSort

Função para classificação simultânea de matrizes array[] e indices[], usando o algoritmo QuickSort.

```
void  MathQuickSort(
   double&  array[],      // matriz de valores
   int&     indices[],    // matriz de índices
   int      first,        // valor inicial
   int      last,         // valor final
   int      mode          // direção
   )
```

Parâmetros

array[]

[in][out] Matriz para classificação.

indices[]

[in][out] Matriz para armazenar índices da matriz de origem.

first

[in] Índice do elemento a partir do qual é necessário começar a classificação.

last

[in] Índice do elemento no qual é necessário acabar a classificação.

mode

[in] Direção da classificação (>0 ascendente, caso contrário, descendente).

© 2000-2024, MetaQuotes Ltd.

MathOrder

Cria uma matiz inteira com um rearranjo na ordem dos elementos da matriz após a classificação.

Versão para trabalhar com uma matriz de valores reais:

```
bool  MathOrder(
   const double&  array[],   // matriz de valores
   int&           result[]   // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores inteiros:

```
bool  MathOrder(
   const int&     array[],   // matriz de valores
   int&           result[]   // matriz de resultado
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz para registro de índices classificados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseNot

Calcula o resultado da operação binária NOT para elementos de matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathBitwiseNot(
   const int&  array[],   // matriz de valores
   int&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathBitwiseNot(
   int&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseAnd

Calcula o resultado da operação binária AND para as matrizes especificadas.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // primeira matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultado
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para registrar resultados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseOr

Calcula o resultado da operação binária OR para as matrizes especificadas.

```
bool  MathBitwiseOr(
   const int&  array1[],  // primeira matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultado
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para registrar resultados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseXor

Calcula o resultado da operação binária XOR para as matrizes especificadas.

```
bool  MathBitwiseXor(
   const int&  array1[],  // primeira matriz de valores
   const int&  array2[],  // segunda matriz de valores
   int&        result[]   // matriz de resultado
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

result[]

[out] Matriz para registrar resultados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseShiftL

Calcula o resultado da operação binária SHL (deslocamento em bits para a esquerda) para elementos de matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathBitwiseShiftL(
   const int&  array[],     // matriz de valores
   const int   n,           // valor do deslocamento
   int&        result[]     // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathBitwiseShiftL(
   int&        array[],     // matriz de valores
   const int   n            // valor do deslocamento
   )
```

Parâmetros

array[]

[in] Matriz de valores.

n

[in] Números de bits para deslocamento.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathBitwiseShiftR

Calcula o resultado da operação binária SHR (deslocamento em bits para a direita) para elementos de matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathBitwiseShiftR(
   const int&  array[],     // matriz de valores
   const int   n,           // valor do deslocamento
   int&        result[]     // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathBitwiseShiftR(
   int&        array[],     // matriz de valores
   const int   n            // valor do deslocamento
   )
```

Parâmetros

array[]

[in] Matriz de valores.

n

[in] Números de bits para deslocamento.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeSum

Gera uma matriz com o valor acumulado.

Versão com registro de resultados numa nova matriz:

```
bool  MathCumulativeSum(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCumulativeSum(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeProduct

Gera uma matriz com o produto acumulado.

Versão com registro de resultados numa nova matriz:

```
bool  MathCumulativeProduct(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCumulativeProduct(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeMin

Gera uma matriz de valores mínimos acumulados.

Versão com registro de resultados numa nova matriz:

```
bool  MathCumulativeMin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCumulativeMin(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeMax

Gera uma matriz de valores máximos acumulados.

Versão com registro de resultados numa nova matriz:

```
bool  MathCumulativeMax(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCumulativeMax(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSin

Calcula o valor da função sin(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathSin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathSin(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCos

Calcula o valor da função cos(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathCos(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCos(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathTan

Calcula o valor da função tan(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathTan(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathTan(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArcsin

Calcula o valor da função arcsin(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArcsin(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArcsin(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArccos

Calcula o valor da função arccos(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArccos(
   const double&   array[],     // matriz de valores
   double&         result[]     // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArccos(
   double&         array[]     // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArctan

Calcula o valor da função arctan(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArctan(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArctan(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSinPi

Calcula o valor da função sin(pi*x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathSinPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathSinPi(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCosPi

Calcula o valor da função cos(pi*x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathCosPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCosPi(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathTanPi

Calcula o valor da função tan(pi*x) para os elementos da matriz.

Versão com registro de resultado numa nova matriz:

```
bool  MathTanPi(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultado na matriz original:

```
bool  MathTanPi(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathAbs

Calcula o valor absoluto dos elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathAbs(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathAbs(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCeil

Retorna o valor de número inteiro mais próximo - desde cima - para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathCeil(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCeil(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathFloor

Retorna o valor de número inteiro mais próximo - desde baixo - para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathFloor(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathFloor(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSqrt

Calcula a raiz quadrada para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathSqrt(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathSqrt(
   double&         array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathExp

Calcula o valor da função exp(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathExp(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathExp(
   double&       array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathPow

Calcula o valor da função pow(x, power) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathPow(
   const double&  array[],    // matriz de valores
   const double   power,      // potência
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathPow(
   double&        array[],    // matriz de valores
   const double   power       // potência
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathLog

Calcula o valor da função log(x) para os elementos da matriz.

Versão para calcular o logaritmo natural com registro de resultados numa nova matriz.

```
bool  MathLog(
   const double&  array[],      // matriz de valores
   double&        result[]      // matriz de resultado
   )
```

Versão para calcular o logaritmo natural com registro de resultados na matriz original.

```
bool  MathLog(
   double&        array[]       // matriz de valores
   )
```

Versão para calcular o logaritmo de acordo com a base especificada com registro de resultados numa nova matriz.

```
bool  MathLog(
   const double&   array[],     // matriz de valores
   const double    base,        // base do logaritmo
   double&         result[]     // matriz de resultado
   )
```

Versão para calcular o logaritmo de acordo com a base especificada com registro de resultados na matriz original.

```
bool  MathLog(
   double&         array[],     // matriz de valores
   const double    base         // base do logaritmo
   )
```

Parâmetros

array[]

[in] Matriz de valores.

base

[in] Base do logaritmo.

array[]

[out] Matriz de valores de saída.

result[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathLog2

Calcula o valor do logaritmo na base 2 para elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathLog2(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathLog2(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathLog10

Calcula o valor do logaritmo na base 10 para elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathLog10(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathLog10(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathLog1p

Calcula o valor da função de log(1+x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathLog1p(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathLog1p(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathDifference

Gera uma matriz com diferenças de elementos y[i]=x[i+lag]-x[i].

Versão para geração única de matrizes de valores reais:

```
bool  MathDifference(
   const double  &array[],     // matriz de valores
   const int     lag,          // latência
   double        &result[]     // matriz de resultado
   )
```

Versão para geração única de matrizes de valores inteiros:

```
bool  MathDifference(
   const int     &array[],     // matriz de valores
   const int     lag,          // latência
   int           &result[]     // matriz de resultado
   )
```

Versão para geração repetida de uma matriz de valores reais (o número de iterações é definido nos parâmetros de entrada):

```
bool  MathDifference(
   const double  &array[],     // matriz de valores
   const int     lag,          // latência
   const int     differences,  // número de iterações
   double        &result[]     // matriz de resultado
   )
```

Versão para geração repetida de uma matriz de valores inteiros (o número de iterações é definido nos parâmetros de entrada):

```
bool  MathDifference(
   const int&    array[],      // matriz de valores
   const int     lag,          // latência
   const int     differences,  // número de iterações
   int&          result[]      // matriz de resultado
   )
```

Parâmetros

array[]

[in] Matriz de valores.

lag

[in] Parâmetro de latência.

differences

[in] Número de iterações.

result[]

© 2000-2024, MetaQuotes Ltd.

[out] Matriz para registrar resultados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSample

Gera uma amostragem aleatória de elementos da matriz.

Versão para trabalhar com uma matriz de valores reais:

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   const int      count,            // quantidade
   double&        result[]          // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores inteiros:

```
bool  MathSample(
   const int&     array[],          // matriz de valores
   const int      count,            // quantidade
   int&           result[]          // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores reais. É possível obter uma amostragem de retorno:

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   const int      count,            // quantidade
   const bool     replace,          // sinalizador
   double&        result[],         // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores inteiros. É possível obter uma amostragem de retorno:

```
bool  MathSample(
   const int&     array[],          // matriz de valores
   const int      count,            // quantidade
   const bool     replace,          // sinalizador
   int&           result[]          // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores reais para os quais são definidas as probabilidades na amostragem.

```
bool  MathSample(
   const double&  array[],          // matriz de valores
   double&        probabilities[],  // matriz de probabilidades
   const int      count,            // quantidade
   double&        result[]          // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores inteiros para os quais são definidas as probabilidades na amostragem.

© 2000-2024, MetaQuotes Ltd.

```
bool  MathSample(
   const int&     array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // quantidade
   int&           result[]           // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores reais para os quais são definidas as probabilidades na amostragem. É possível obter uma amostragem de retorno:

```
bool  MathSample(
   const double&  array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // quantidade
   const bool     replace,           // sinalizador
   double&        result[]           // matriz de resultado
   )
```

Versão para trabalhar com uma matriz de valores inteiros para os quais são definidas as probabilidades na amostragem. É possível obter uma amostragem de retorno:

```
bool  MathSample(
   const int&     array[],           // matriz de valores
   double&        probabilities[],   // matriz de probabilidades
   const int      count,             // quantidade
   const bool     replace,           // sinalizador
   int&           result[]           // matriz de resultado
   )
```

Parâmetros

array[]

[in] Matriz de valores inteiros.

probabilities[]

[in] Matriz de probabilidades com as quais é realizada a amostragem de elementos.

count

[in] Número de elementos.

replace

[in] Parâmetro que permite realizar a seleção com retorno.

result[]

[out] Matriz para registro de resultados.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Observação

© 2000-2024, MetaQuotes Ltd.

O argumento replace=true permite realizar a seleção aleatória com seu retorno para o conjunto inicial.

© 2000-2024, MetaQuotes Ltd.

MathTukeySummary

Calcula o resumo de cinco números de Tukey (mínimo, quartil inferior, média, quartil superior, máximo) para os elementos da matriz.

```
bool  MathTukeySummary(
   const double&  array[],       // matriz de valores
   const bool     removeNAN,     // sinalizador
   double&        minimum,       // valor mínimo
   double&        lower_hinge,   // quartil inferior
   double&        median,        // valor médio
   double&        upper_hinge,   // quartil superior
   double&        maximum        // valor máximo
   )
```

Parâmetros

array[]

[in] Matriz de valores reais.

removeNAN

[in] Sinalizador que indica se é necessário apagar os valores não numéricos.

minimum

[out] Variável para registro do valor mínimo.

lower_hinge

[out] Variável para registro do quartil inferior.

median

[out] Variável para registro do valor médio.

upper_hinge

[out] Variável para registro do quartil superior.

maximum

[out] Variável para registro do valor máximo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathRange

Calcula os valores mínimos e máximos dos elementos da matriz.

```
bool  MathRange(
   const double&  array[],  // matriz de valores
   double&        min,      // valor mínimo
   double&        max       // valor máximo
   )
```

Parâmetros

array[]

[in] Matriz de valores.

min

[out] Variável para registro do valor mínimo.

max

[out] Variável para registro do valor máximo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathMin

Retorna o valor mínimo de todos os elementos na matriz.

```
double  MathMin(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Valor mínimo.

© 2000-2024, MetaQuotes Ltd.

MathMax

Retorna o valor máximo de todos os elementos da matriz.

```
double  MathMax(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Valor máximo.

© 2000-2024, MetaQuotes Ltd.

MathSum

Retorna a soma dos elementos da matriz.

```
double  MathSum(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Soma de elementos.

© 2000-2024, MetaQuotes Ltd.

MathProduct

Retorna o produto dos elementos da matriz.

```
double  MathProduct(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Produto de elementos.

© 2000-2024, MetaQuotes Ltd.

MathStandardDeviation

Calcula o desvio padrão dos elementos da matriz.

```
double  MathStandardDeviation(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Desvio padrão.

© 2000-2024, MetaQuotes Ltd.

MathAverageDeviation

A função calcula o desvio médio dos elementos na matriz.

```
double  MathAverageDeviation(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Desvio médio dos elementos da matriz.

© 2000-2024, MetaQuotes Ltd.

MathMedian

Calcula o valor de mediana dos elementos da matriz.

```
double  MathMedian(
   double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Valor de mediana.

© 2000-2024, MetaQuotes Ltd.

MathMean

Calcula o valor médio dos elementos da matriz.

```
double  MathMean(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Valor médio.

© 2000-2024, MetaQuotes Ltd.

MathVariance

A função calcula a dispersão (segundo momento) dos elementos da matriz.

```
double  MathVariance(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Valor da dispersão.

© 2000-2024, MetaQuotes Ltd.

MathSkewness

A função calcula a obliquidade dos elementos da matriz (terceiro momento).

```
double  MathSkewness(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Obliquidade.

© 2000-2024, MetaQuotes Ltd.

MathKurtosis

Calcula a curtose (quarto momento) dos elementos da matriz.

```
double  MathKurtosis(
   const double&  array[]   // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

Valor de retorno

Coeficiente de curtose.

© 2000-2024, MetaQuotes Ltd.

MathExpm1

Calcula o valor da função exp(x)-1 para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathExpm1(
   const double&   array[],   // matriz de valores
   double&         result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathExpm1(
   double&         array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSinh

Calcula o valor da função sinh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathSinh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathSinh(
   double&       array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCosh

Calcula o valor da função cosh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathCosh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathCosh(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathTanh

Calcula o valor da função tanh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathTanh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathTanh(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArcsinh

Calcula o valor da função arcsinh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArcsinh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArcsinh(
   double&         array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArccosh

Calcula o valor da função arccosh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArccosh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArccosh(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathArctanh

Calcula o valor da função arctanh(x) para os elementos da matriz.

Versão com registro de resultados numa nova matriz:

```
bool  MathArctanh(
   const double&  array[],   // matriz de valores
   double&        result[]   // matriz de resultado
   )
```

Versão com registro de resultados na matriz original:

```
bool  MathArctanh(
   double&        array[]    // matriz de valores
   )
```

Parâmetros

array[]

[in] Matriz de valores.

result[]

[out] Matriz de valores de saída.

array[]

[out] Matriz de valores de saída.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathSignif

Arredonda um valor para o número especificado de dígitos na mantissa.

Versão para trabalhar com valores reais:

```
double  MathSignif(
   const double   x,          // valor
   const int      digits      // número de dígitos
   )
```

Valor de retorno

Valor arredondado.

Versão para trabalhar com uma matriz de valores reais com registro de resultados numa matriz separada:

```
bool  MathSignif(
   const double&  array[],    // matriz de valores
   int            digits,     // número de dígitos
   double         result[]    // matriz de resultado
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Versão para trabalhar com uma matriz de valores reais com registro de resultados na matriz de origem:

```
bool  MathSignif(
   double&        array[],    // matriz de valores
   int            digits      // número de dígitos
   )
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

Parâmetros

x

[in] Valor real para arredondamento.

digits

[in] Número de dígitos.

array[]

[in] Matriz de valores reais.

array[]

[out] Matriz de valores de saída.

result[]

© 2000-2024, MetaQuotes Ltd.

[out] Matriz de valores de saída.

© 2000-2024, MetaQuotes Ltd.

MathRank

Calcula a posição dos elementos da matriz.

Versão para trabalhar com uma matriz de valores reais:

```
bool  MathRank(
   const double&  array[],  // matriz de valores
   double&        rank[]    // matriz de rankings
   )
```

Versão para trabalhar com uma matriz de valores inteiros:

```
bool  MathRank(
   const int&     array[],  // matriz de valores
   double&        rank[]    // matriz de rankings
   )
```

Parâmetros

array[]

[in] Matriz de valores.

rank[]

[out] Matriz para registro de rankings.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationPearson

Calcula o coeficiente de correlação de Pearson.

Versão para trabalhar com matrizes de valores reais:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // primeira matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlação
   )
```

Versão para trabalhar com matrizes de valores inteiros:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // primeira matriz de valores
   const int&     array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlação
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

r

[out] Variável para registro do coeficiente de correlação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationSpearman

Calcula o coeficiente de correlação de Spearman.

Versão para trabalhar com matrizes de valores reais:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // primeira matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        r          // coeficiente de correlação
   )
```

Versão para trabalhar com matrizes de valores inteiros:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // primeira matriz de valores
   const int&     array2[],   // segunda matriz de valores
   double&        r           // coeficiente de correlação
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

r

[out] Variável para registro do coeficiente de correlação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCorrelationKendall

Calcula o coeficiente de correlação de Kendall.

Versão para trabalhar com matrizes de valores reais:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // primeira matriz de valores
   const double&  array2[],  // segunda matriz de valores
   double&        tau        // coeficiente de correlação
   )
```

Versão para trabalhar com matrizes de valores inteiros:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // primeira matriz de valores
   const int&     array2[],  // segunda matriz de valores
   double&        tau        // coeficiente de correlação
   )
```

Parâmetros

array1[]

[in] Primeira matriz de valores.

array2[]

[in] Segunda matriz de valores.

tau

[out] Variável para registro do coeficiente de correlação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathQuantile

Calcula o quantis seletivos que corresponde às probabilidades definidas: Q[i](p) = (1 - gamma) *x[j] + gamma*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // matriz de valores
   const double&  probs[],     // matriz de probabilidades
   double&        quantile[]   // matriz para registro de quantis
   )
```

Parâmetros

array[]

[in] Matriz de valores.

probs[]

[in] Matriz de probabilidades.

quantile[]

[out] Matriz para registro de quantis.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

A função calcula a função densidade de probabilidade empírica (pdf) para os valores aleatórios.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // matriz de valores aleatórios
   const int      count,     // número de pares
   double&        x[],       // matriz de valores x
   double&        pdf[]      // matriz de valores pdf
   )
```

Parâmetros

array[]

[in] Matriz de valores aleatórios.

count

[in] Número de pares (x, pdf(x)).

x[]

[out] Matriz para registro de valores x.

pdf[]

[out] Matriz para registro de valores pdf(x).

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

Calcula a distribuição cumulativa empírica (cdf) para os valores aleatórios da matriz.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],   // matriz de valores aleatórios
   const int      count,     // número de pares
   double&        x[],       // matriz de valores x
   double&        cdf[]      // matriz de valores cdf
   )
```

Parâmetros

array[]

[in] Matriz de valores aleatórios.

count

[in] Número de pares (x, cdf(x)).

x[]

[out] Matriz para registro de valores x.

cdf[]

[out] Matriz para registro de valores cdf(x).

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

A Fuzzy consiste na biblioteca para trabalhar com lógica difusa

A **lógica difusa** é uma generalização da lógica aristotélica tradicional para o caso quando a verdade é vista como uma variável linguística. Como na lógica clássica, para a lógica difusa, são definidas suas operações de lógica difusa sobre os conjuntos difusos. Para os conjuntos difusos, existem todas as mesmas operações que para os conjuntos convencionais, com a particularidade de serem mais complexas. Observe que a composição dos conjuntos difusos também abrange um conjunto difuso.

A aproximação real ao refletir a realidade e um elevado nível de subjetividade que pode levar a erros significativos nos resultados dos cálculos durante seu uso são uma das principais características da lógica difusa que a distinguem da clássica.

O modelo (ou sistema) difuso é um modelo matemático, cuja base de cálculo é o fundamento da lógica difusa. Recorre-se à plotagem de tais modelos em caso de o objeto de estudo ter uma formalização muito fraca e sua descrição matemática ser muito complicada ou desconhecida. A qualidade das saídas desses modelos (erro de modelo) depende diretamente de perito que elaborou e definir o modelo. Para minimizar os erros, a melhor opção consiste em compilar o modelo mais completo e abrangente e, em seguida, configurá-lo usando as ferramentas de aprendizado de máquina, numa grande amostragem de treinamento.

O progresso na plotagem do modelo pode ser dividida em três fases principais:

1. Determinação dos parâmetros de entrada e de saída do modelo.
2. Construção de uma base de conhecimento.
3. Seleção de um dos métodos de inferência difusa (Mamdani ou Sugeno ou Sugeno).

Os outros dois dependem da primeira fase, uma vez que ela define o funcionamento futuro do modelo. A base de conhecimentos ou, como é também é chamada, a base de regras abrange o conjunto de regras difusas do tipo: "se, então" que definem a relação entre as entradas e saídas do objeto pesquisado. No sistema, o número de regras é determinado pelo Expert Advisor e também não é restrito. Formato generalizado de regras difusas:

Se há condição (envio) de regra, então há conclusão quanto à regra.

Se a condição da regra caracteriza o estado atual do objeto, a conclusão é como essa condição irá afetar o objeto. A visão geral das condições e conclusões não pode ser separada, uma vez que eles são determinados pela inferência difusa.

Cada regra tem um peso no sistema, é este parâmetro que caracteriza a importância da regra no modelo. Os fatores de ponderação são atribuídos à regra no intervalo [0, 1]. Em muitos dos exemplos sobre modelos difusos, que podem ser encontrados na literatura, os dados de peso não são especificados, mas isso não quer dizer que eles não existam, por outro lado, na verdade, para cada regra a partir da base de dados, em tal caso, o peso é fixo e igual a um. As condições e conclusões de cada regra podem ser de dois tipos:

1. simples — nela está envolvida uma variável difusa;
2. composta — participam várias variáveis difusas.

O sistema de inferência difuso é determinado dependendo da base de conhecimento criada para o modelo. Inferência difusa é chamada a obtenção de uma conclusão na forma de conjunto difuso

© 2000-2024, MetaQuotes Ltd.

correspondente aos valores atuais das entradas com uso de uma base difusa de conhecimentos e operações difusas. Há dos tipos básicos de inferência difusa, isto é, Mamdani ou Sugeno.

© 2000-2024, MetaQuotes Ltd.

Funções de associação

Designa-se de**função de associação** *(membership function)* à função que permite calcular o grau de pertinência de um elemento arbitrário do conjunto universal do conjunto difuso. Consequentemente, em todo o domínio, a função de associação dever ter valores no intervalo [0, 1].

Na maioria dos casos, a função de associação é monótona e contínua.

Classe de funções de associação	Descrição
CConstantMembershipFunction	Classe para implementar uma função de associação na forma de uma reta paralela aos eixos coordenados
CCompositeMembershipFunction	Classe para implementar a composição de funções de associação
CDifferencTwoSigmoidalMembershipFunction	Classe para implementar uma função de associação na forma da diferença entre duas funções sigmóides com parâmetros A1, A2, C1, C2
CGeneralizedBellShapedMembershipFunction	Classe para implementar a função de associação generalizada em forma de sino com parâmetros A, B e C
CNormalCombinationMembershipFunction	Classe para implementar uma função de associação de Gauss dupla com parâmetros B1, B2, Sigma1 e Sigma2
CNormalMembershipFunction	Classe para implementar uma função de associação simétrica de Gauss com parâmetros B e Sigma
CP_ShapedMembershipFunction	Classe para implementar a função de associação em forma de Pi com parâmetros A, B, C e D
CProductTwoSigmoidalMembershipFunction	Classe para implementar uma função de associação na forma do produto entre duas funções sigmóides com parâmetros A1, A2, C1, C2
CS_ShapedMembershipFunction	Classe para implementar funções de associação em forma de S com parâmetros A e B
CTrapezoidMembershipFunction	Classe para implementar uma função de associação trapezoidal com parâmetros X1, X2, X3 e X4
CTriangularMembershipFunction	Classe para implementar uma função de associação triangular com parâmetros X1, X2, X3

© 2000-2024, MetaQuotes Ltd.

Classe de funções de associação	Descrição
CSigmoidalMembershipFunction	Classe para implementar uma função de associação sigmóide com parâmetros A e C
CZ_ShapedMembershipFunction	Classe para implementar funções de associação em forma de Z com parâmetros A e B.
IMembershipFunction	Classe base para todas as classes da função de associação.

© 2000-2024, MetaQuotes Ltd.

CConstantMembershipFunction

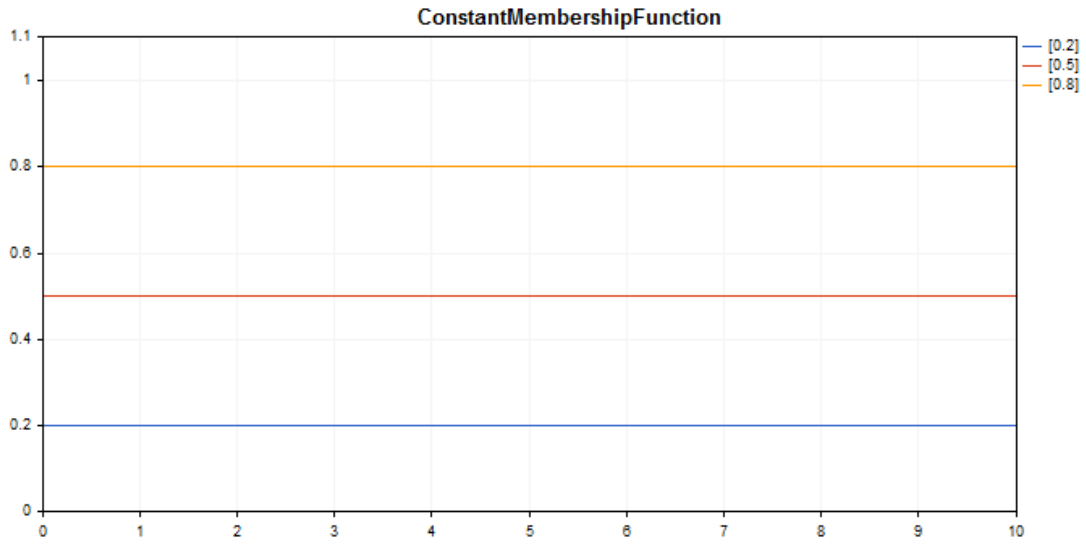
Classe para implementar uma função de associação na forma de uma reta paralela aos eixos coordenados.

Descrição

A função é descrita pela equação:

y(x)=c

Consequentemente, o grau de pertinência para esta função é a mesma em todo o eixo real e é igual aos parâmetros especificados no construtor.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
IMembershipFunction
CConstantMembershipFunction

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                   ConstantMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- Create wrappers for membership functions
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const double  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

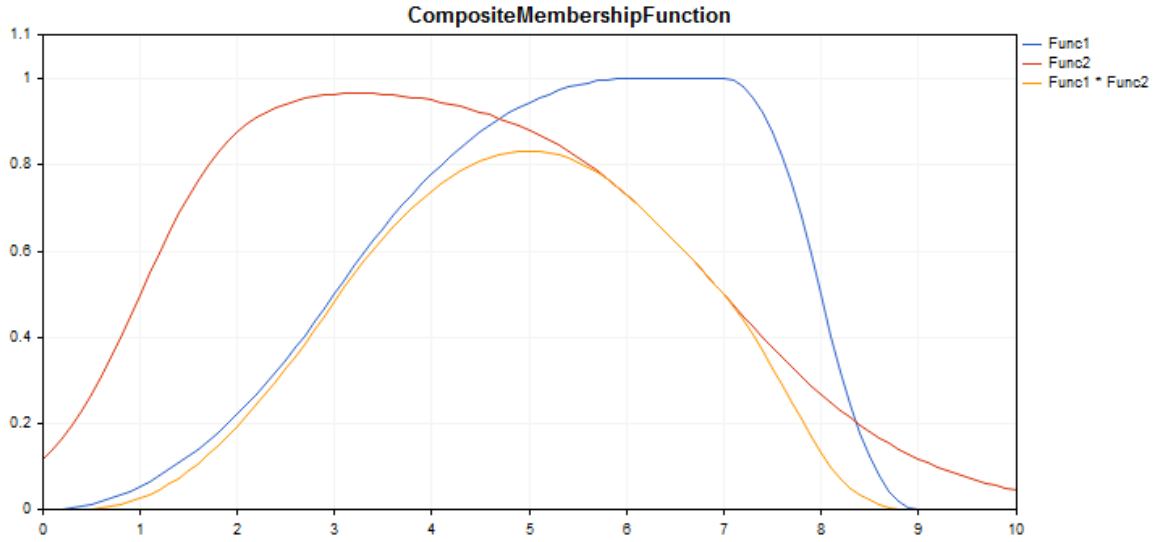
© 2000-2024, MetaQuotes Ltd.

CCompositeMembershipFunction

Classe para implementar a composição de funções de associação.

Descrição

A composição das funções de associação consiste na união de duas ou mais funções de associação usando o operador especificado.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CCompositeMembershipFunction

Métodos de classe

Método de classe	Descrição
CompositionType	Define o operador de composição.
MembershipFunctions	Retorna a lista de funções de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- Create wrappers for membership functions
double ProductTwoSigmoidalMembershipFunctions(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,"F
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 * Func
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
```

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Define o operador de composição.

```
void  CompositionType(
   MfCompositionType  value      // tipo de operador
   )
```

Parâmetros

value

[in] Tipo de operador de composição.

Observação

Estão disponíveis os seguintes tipos de operadores:

- MinMF (mínimo de funções)
- MaxMF (máximo de funções)
- ProdMF (produto de funções)
- SumMF (soma de funções)

MembershipFunctions

Retorna a lista de funções de associação incluídas na composição.

```
CList*  MembershipFunctions(
   void        // lista de funções de associação
   )
```

Valor de retorno

Lista de funções de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

© 2000-2024, MetaQuotes Ltd.

Valor da função de associação.

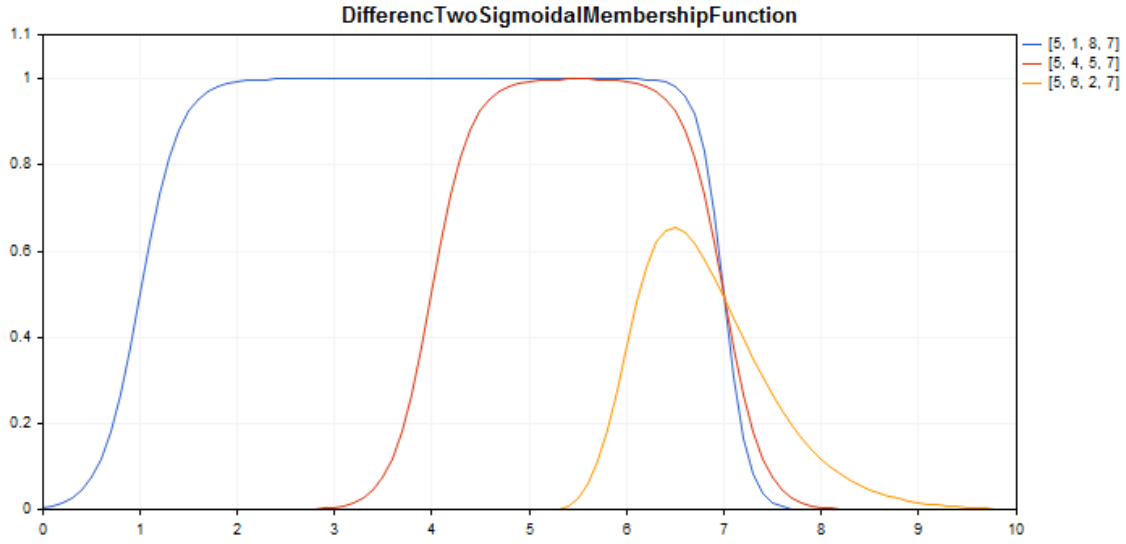
© 2000-2024, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

Classe para implementar uma função de associação na forma da diferença entre duas funções sigmóides com parâmetros A1, A2, C1, C2.

Descrição

A função é baseada na curva sigmóide. Você pode usá-la para criar uma função de associação com valores iguais a 1, começando com um determinado valor do argumento. Estas funções são adequados, se é necessário definir os termos linguísticos de tipo "curto" ou "longo".



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CDifferencTwoSigmoidalMembershipFunction

Métodos de classe

Método de classe	Descrição
A1	Retorna e define o coeficiente de inclinação da primeira função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
A2	Retorna e define o coeficiente de inclinação da segunda função de associação.
C1	Retorna o parâmetro de coordenada da inflexão da primeira função de associação.
C2	Retorna o parâmetro de coordenada da inflexão da segunda função de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- Create wrappers for membership functions
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembership Function2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
```

```
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Método Get)

Retorna o coeficiente de inclinação da primeira função de associação.

```
double  A1()
```

Valor de retorno

Valor do coeficiente de inclinação.

A1 (Método Set)

Define o coeficiente de inclinação da primeira função de associação.

```
void  A1(
   const double  a1      // valor do coeficiente de inclinação
   )
```

Parâmetros

a1

[in] Valor do coeficiente de inclinação.

A2 (Método Get)

Retorna o coeficiente de inclinação da segunda função de associação.

```
double  A2()
```

Valor de retorno

Valor do coeficiente de inclinação.

A2 (Método Set)

© 2000-2024, MetaQuotes Ltd.

Define o coeficiente de inclinação da segunda função de associação.

```
void  A2(
   const double  a2      // valor do coeficiente de inclinação
   )
```

Parâmetros

a2

[in] Valor do coeficiente de inclinação.

C1 (Método Get)

Retorna o parâmetro de coordenada da inflexão da primeira função de associação.

```
double  C1()
```

Valor de retorno

Valor da coordenada de inflexão.

C1 (Método Set)

Define o parâmetro de coordenada da inflexão da primeira função de associação.

```
void  C1(
   const double  c1      // valor da coordenada de inflexão
   )
```

Parâmetros

c1

[in] Valor de coordenada de inflexão.

C2 (Método Get)

Retorna o parâmetro de coordenada da inflexão da segunda função de associação.

```
double  C2()
```

Valor de retorno

Valor da coordenada de inflexão.

C2 (Método Set)

Define o parâmetro de coordenada da inflexão da segunda função de associação.

```
void  C2(
   const double  c2      // valor de coordenada de inflexão
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

c2

[in] Valor de coordenada de inflexão.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const double  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

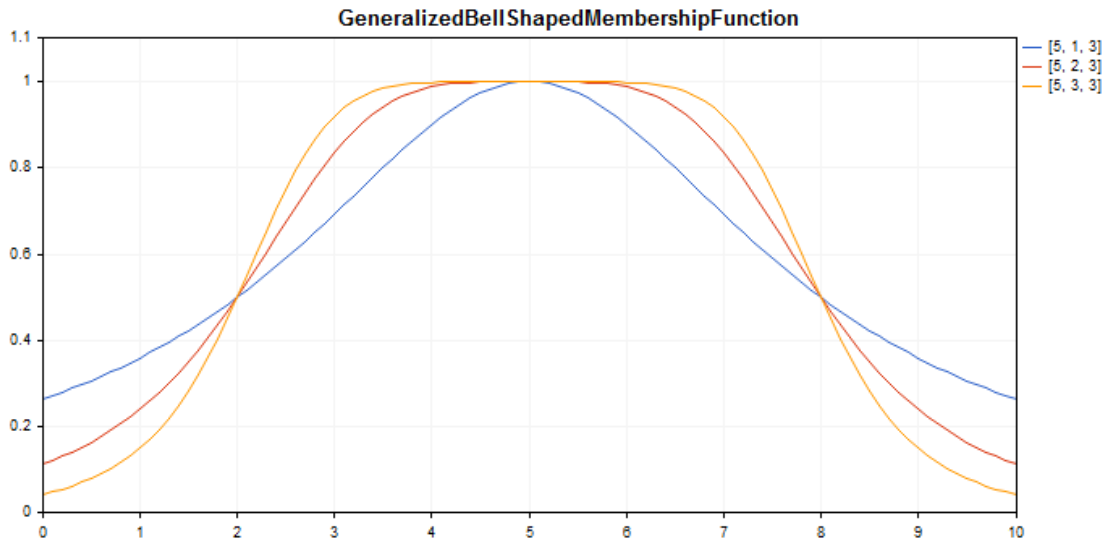
© 2000-2024, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

Classe para implementar a função de associação generalizada em forma de sino com parâmetros A, B e C.

Descrição

A função de associação generalizada tem forma de sino, semelhante à função de Gauss. Em todo o domínio, a função é suave e tem valores diferentes de zero.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CGeneralizedBellShapedMembershipFunction

Métodos de classe

Método de classe	Descrição
A	Retorna e define o coeficiente de concentração da função de associação.
B	Retorna e define o coeficiente de inclinação da função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
C	Retorna e define a coordenada do máximo da função de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- Create wrappers for membership functions
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna o coeficiente de concentração da função de associação.

```
double  A()
```

Valor de retorno

Valor do coeficiente de concentração.

A (Método Set)

Define o coeficiente de concentração da função de associação.

```
void  A(
   const double  a      // valor do coeficiente de concentração
   )
```

Parâmetros

a

[in] Coeficiente de concentração da função de associação.

B (Método Get)

Retorna o coeficiente de inclinação da função de associação.

```
double  B()
```

Valor de retorno

Valor do coeficiente de inclinação.

B (Método Set)

Define o coeficiente de inclinação da função de associação.

© 2000-2024, MetaQuotes Ltd.

```
void B(
   const double b      // valor do coeficiente de inclinação
   )
```

Parâmetros

b

[in] Coeficiente de inclinação da função de associação.

C (Método Get)

Retorna coordenada do máximo da função de associação.

```
double C()
```

Valor de retorno

Coordenada do máximo da função de associação.

C (Método Set)

Define a coordenada do máximo da função de associação.

```
void C(
   const double c      // valor da coordenada do máximo
   )
```

Parâmetros

c

[in] Coordenada do máximo da função de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double GetValue(
   const x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

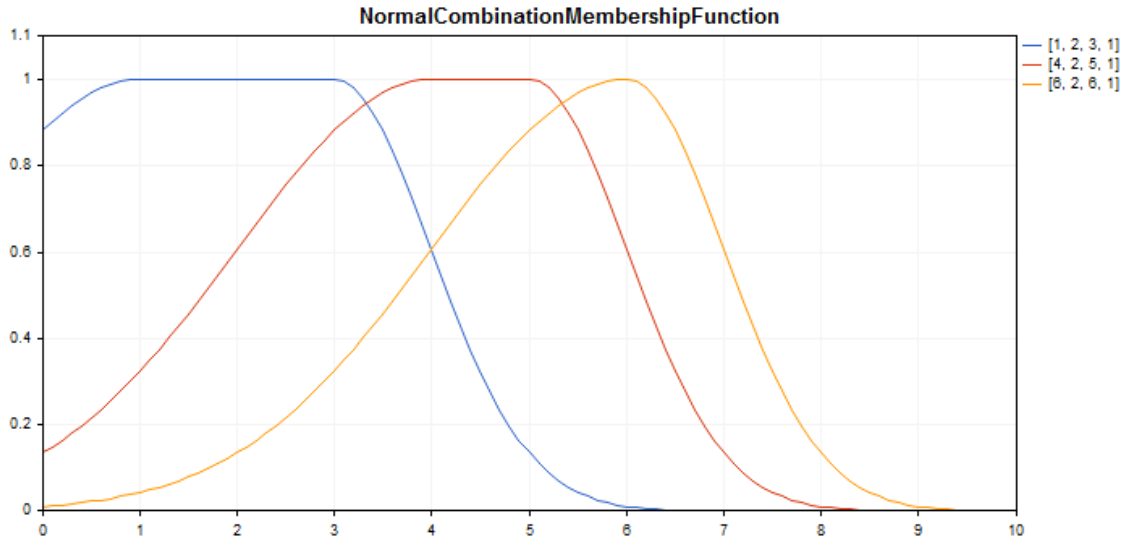
© 2000-2024, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

Classe para implementar uma função de associação de Gauss dupla com parâmetros B1, B2, Sigma1 e Sigma2.

Descrição

A função de associação de Gauss dupla é gerada utilizando distribuição de Gauss. Ele permite definir funções de associação assimétricas. Em todo o domínio, a função é suave e tem valores diferentes de zero.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CNormalCombinationMembershipFunction

Métodos de classe

Método de classe	Descrição
B1	Retorna e define o valor do primeiro centro da função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
B2	Retorna e define o valor do segundo centro da função de associação.
Sigma1	Retorna e define o primeiro parâmetro da curvatura da função de associação.
Sigma2	Retorna e define o segundo parâmetro da curvatura da função de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- Create wrappers for membership functions
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[1,
```

```
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[4,
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[6,
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (Método Get)

Retorna o valor do primeiro centro da função de associação.

```
double  B1()
```

Valor de retorno

Valor do primeiro centro da função de associação.

B1 (Método Set)

Define o valor do primeiro centro da função de associação.

```
void  B1(
   const double  b1      // valor do primeiro centro da função de associação
   )
```

Parâmetros

b

[in] Valor do primeiro centro da função de associação.

B2 (Método Get)

Retorna o valor do segundo centro da função de associação.

```
double  B2()
```

Valor de retorno

Valor do segundo centro da função de associação.

© 2000-2024, MetaQuotes Ltd.

B2 (Método Set)

Define o valor do segundo centro da função de associação.

```
void  B2(
   const double  b2      // valor do segundo centro da função de associação
   )
```

Parâmetros

b2

[in] Valor do segundo centro da função de associação.

Sigma1 (Método Get)

Retorna o primeiro parâmetro da curvatura da função de associação.

```
double  Sigma1()
```

Valor de retorno

Primeiro parâmetro da curvatura da função de associação.

Sigma1 (Método Set)

Define o primeiro parâmetro da curvatura da função de associação.

```
void  Sigma1(
   const double  sigma1      // valor do primeiro parâmetro da curvatura
   )
```

Parâmetros

sigma1

[in] Primeiro parâmetro da curvatura da função de associação.

Sigma2 (Método Get)

Retorna o segundo parâmetro da curvatura da função de associação.

```
double  Sigma2()
```

Valor de retorno

Valor do segundo parâmetro da curvatura da função de associação.

Sigma2 (Método Set)

© 2000-2024, MetaQuotes Ltd.

Define o segundo parâmetro da curvatura da função de associação.

```
void  Sigma2(
   const double  sigma2      // valor do segundo parâmetro da curvatura
   )
```

Parâmetros

sigma2

[in] Segundo parâmetro da curvatura da função de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

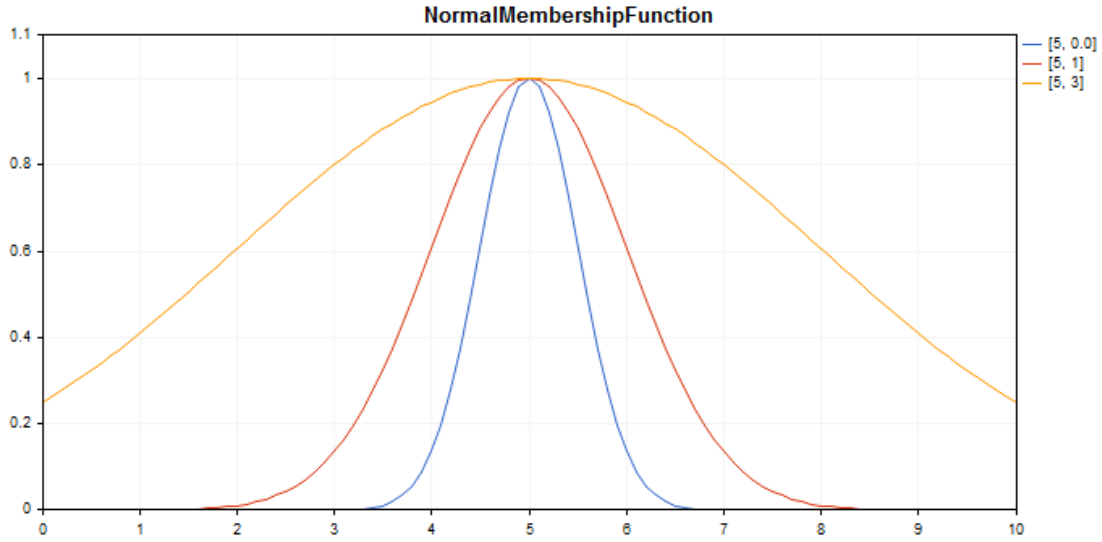
© 2000-2024, MetaQuotes Ltd.

CNormalMembershipFunction

Classe para implementar uma função de associação simétrica de Gauss com parâmetros B e Sigma.

Descrição

A função de associação simétrica de Gauss é gerada utilizando distribuição de Gauss. Em todo o domínio, a função é suave e tem valores diferentes de zero.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CNormalMembershipFunction

Métodos de classe

Método de classe	Descrição
B	Retorna e define o centro da função de associação.
Sigma	Retorna e define o parâmetro da curvatura da função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                   NormalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- Create wrappers for membership functions
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (Método Get)

Retorna o centro da função de associação.

```
double  B()
```

Valor de retorno

Valor de centro da função de associação.

B (Método Set)

Define o valor de centro da função de associação.

```
void  B(
   const double  b      // valor de centro da função
   )
```

Parâmetros

b

[in] Valor de centro da função de associação.

Sigma (Método Get)

Retorna o parâmetro da curvatura da função de associação.

```
double  Sigma()
```

Valor de retorno

Valor do parâmetro da curvatura da função de associação.

Sigma (Método Set)

Define o valor do parâmetro da curvatura da função de associação.

```
void  Sigma(
   const double  sigma      // valor do parâmetro da curvatura
   )
```

Parâmetros

sigma

© 2000-2024, MetaQuotes Ltd.

[in] Parâmetro da curvatura da função de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const double  x      // argumento
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

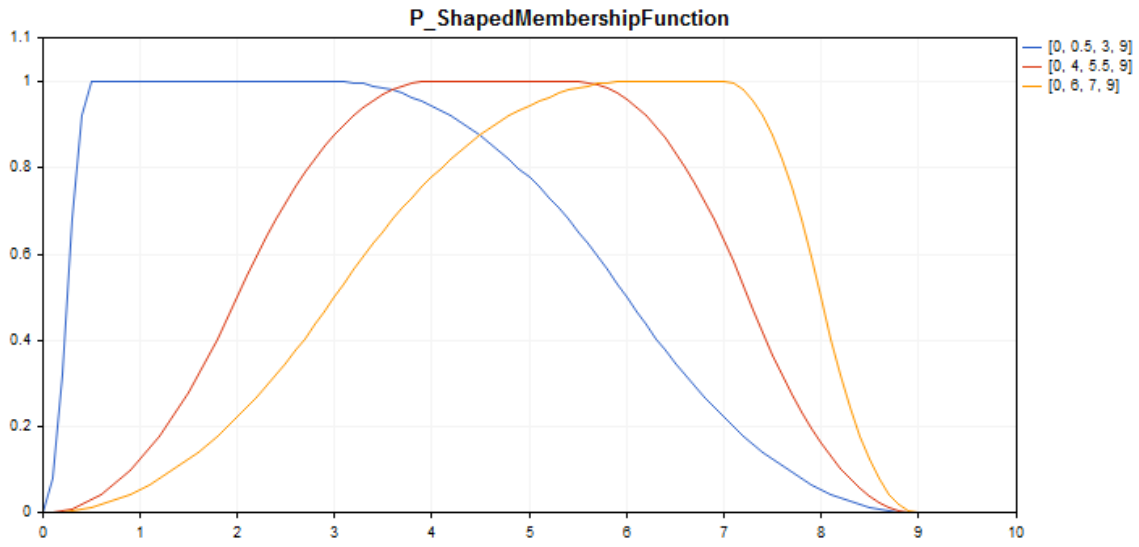
© 2000-2024, MetaQuotes Ltd.

CP_ShapedMembershipFunction

Classe para implementar a função de associação em forma de Pi com parâmetros A, B, C e D.

Descrição

A função de associação em forma de Pi tem forma de trapézio curvilíneo. A função é usada para definir funções de associação assimétricas com uma transição suave de avaliação pessimista para optimista do número difuso.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CP_ShapedMembershipFunction

Métodos de classe

Método de classe	Descrição
A	Retorna e define o parâmetro de início do conjunto difuso.
B	Retorna e define o primeiro parâmetro de núcleo do conjunto difuso.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
C	Retorna e define o segundo parâmetro de núcleo do conjunto difuso.
D	Retorna e define o parâmetro de fim do conjunto difuso.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                  P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembership Function func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- Create wrappers for membership functions
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5, 3,
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4, 5.5,
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7, 9
//--- sets the X-axis properties
```

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna o parâmetro de início do conjunto difuso.

```
double  A()
```

Valor de retorno

Parâmetro de início do conjunto difuso.

A (Método Set)

Define o parâmetro de início do conjunto difuso.

```
void  A(
   const double  a      // parâmetro de início do conjunto difuso
   )
```

Parâmetros

a

[in] Parâmetro de início do conjunto difuso.

B (Método Get)

Retorna o primeiro parâmetro de núcleo do conjunto difuso.

```
double  B()
```

Valor de retorno

Primeiro parâmetro de núcleo do conjunto difuso.

B (Método Set)

Define o primeiro parâmetro de núcleo do conjunto difuso.

```
void  B(
   const double  b      // valor do primeiro parâmetro de núcleo do conjunto difuso
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

b

[in] Primeiro parâmetro de núcleo do conjunto difuso.

C (Método Get)

Retorna o segundo parâmetro de núcleo do conjunto difuso.

```
double  C()
```

Valor de retorno

Segundo parâmetro de núcleo do conjunto difuso.

C (Método Set)

Define o segundo parâmetro de núcleo do conjunto difuso.

```
void  C(
   const double  c      // valor do segundo parâmetro de núcleo do conjunto difuso
   )
```

Parâmetros

c

[in] Segundo parâmetro de núcleo do conjunto difuso.

D (Método Get)

Retorna o parâmetro de fim do conjunto difuso.

```
double  D()
```

Valor de retorno

Valor do parâmetro de fim do conjunto difuso.

D (Método Set)

Retorna o parâmetro de fim do conjunto difuso.

```
void  D(
   const double  d      // valor do parâmetro de fim do conjunto difuso
   )
```

Parâmetros

d

[in] Valor do parâmetro de fim do conjunto difuso.

© 2000-2024, MetaQuotes Ltd.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const double  x
   )
```

Parâmetros

x

[in] argumento de associação

Valor de retorno

Valor da função de associação.

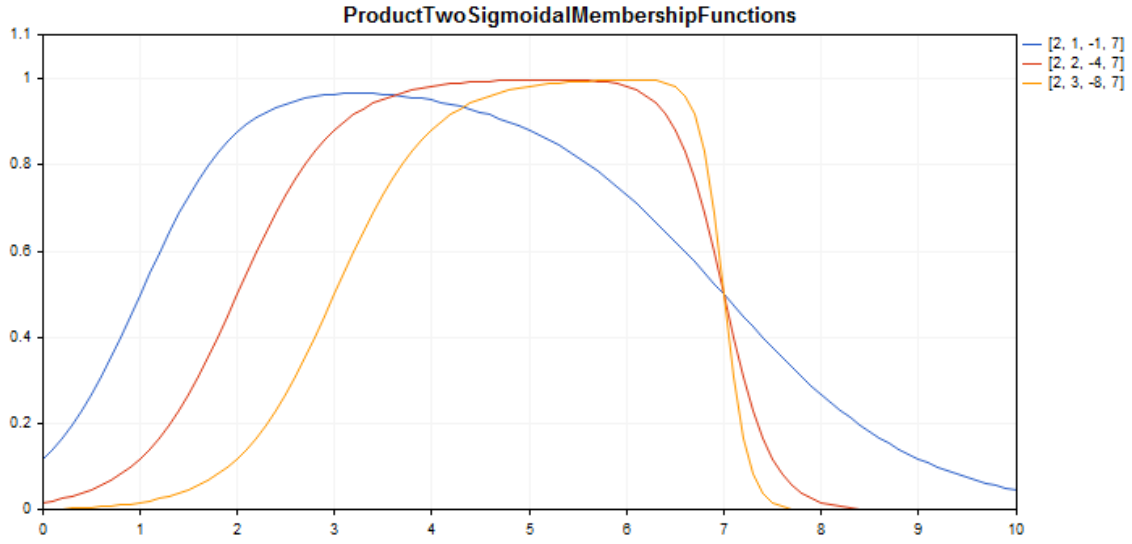
© 2000-2024, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

Classe para implementar uma função de associação na forma do produto entre duas funções sigmóides com parâmetros A1, A2, C1, C2.

Descrição

O produto de duas funções de associação sigmóides é usado para definir as funções assimétricas suaves. Você pode usá-lo para criar uma função de associação com valores iguais a 1, começando com um determinado valor do argumento. Estas funções são adequados, se é necessário definir os termos linguísticos de tipo "curto" ou "longo".



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CProductTwoSigmoidalMembershipFunctions

Métodos de classe

Método de classe	Descrição
A1	Retorna e define o coeficiente de inclinação da primeira função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
A2	Retorna e define o coeficiente de inclinação da segunda função de associação.
C1	Retorna o parâmetro de coordenada da inflexão da primeira função de associação.
C2	Retorna o parâmetro de coordenada da inflexão da segunda função de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- Create wrappers for membership functions
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
```

```
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES,'
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Método Get)

Retorna o coeficiente de inclinação da primeira função de associação.

```
double  A1()
```

Valor de retorno

Coeficiente de inclinação da primeira função de associação.

A1 (Método Set)

Define o coeficiente de inclinação da primeira função de associação.

```
void  A1(
   const double  a1      // coeficiente de inclinação da primeira função de associaçã
   )
```

Parâmetros

a1

[in] Coeficiente de inclinação da primeira função de associação.

A2 (Método Get)

Retorna o coeficiente de inclinação da segunda função de associação.

```
double  A2()
```

Valor de retorno

Coeficiente de inclinação da segunda função de associação.

© 2000-2024, MetaQuotes Ltd.

A2 (Método Set)

Define o coeficiente de inclinação da segunda função de associação.

```
void  A2(
   const double  a2      // coeficiente de inclinação da segunda função de associação
   )
```

Parâmetros

a2

[in] Coeficiente de inclinação da segunda função de associação.

C1 (Método Get)

Retorna o parâmetro de coordenada da inflexão da primeira função de associação.

```
double  C1()
```

Valor de retorno

Coordenada de inflexão da primeira função de associação.

C1 (Método Set)

Define a coordenada de inflexão da primeira função de associação.

```
void  C1(
   const double  c1      // coordenada de inflexão da primeira função de associação
   )
```

Parâmetros

c1

[in] Coordenada de inflexão da primeira função de associação.

C2 (Método Get)

Retorna o parâmetro de coordenada da inflexão da segunda função de associação.

```
double  C2()
```

Valor de retorno

Coordenada de inflexão da segunda função de associação.

C2 (Método Set)

Define a coordenada de inflexão da segunda função de associação.

```
void  C2(
   const double  c2      // coordenada de inflexão da segunda função de associação
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

c2

[in] Coordenada de inflexão da segunda função de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

© 2000-2024, MetaQuotes Ltd.

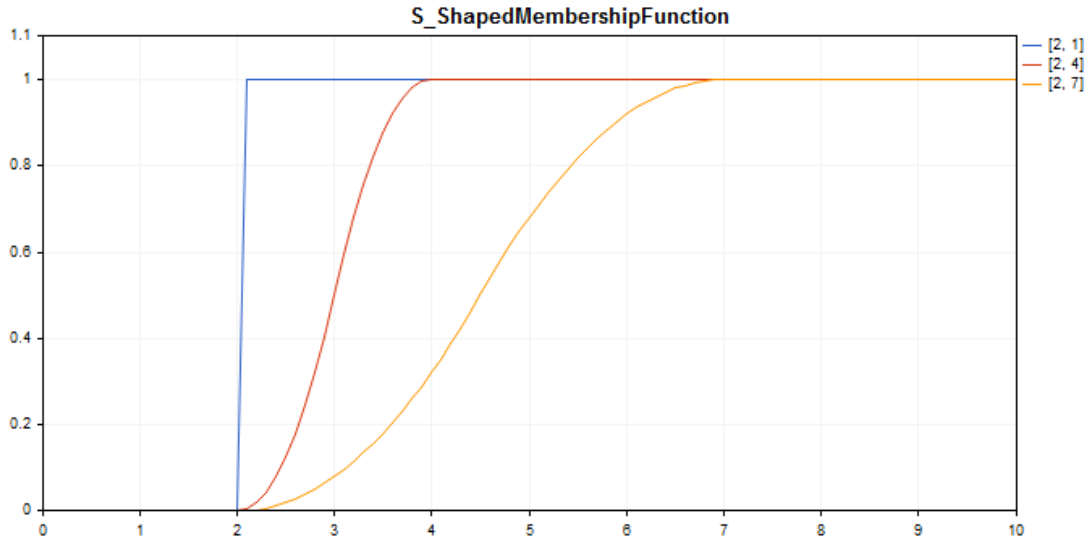
CS_ShapedMembershipFunction

Classe para implementar funções de associação em forma de S com parâmetros A e B.

Descrição

A função define a função de associação em forma de S de dois parâmetros. Trata-se de uma função de associação decrescente que aceita valores 1a 0. Os parâmetros A e B da função definem o intervalo dentro do qual a função não linear diminui de 0 a 1.

Usando a função são representados conjuntos difusos de tipo "muito alto" (ou seja, são definidas funções de associação decrescentes com saturação).



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CS_ShapedMembershipFunction

Métodos de classe

Método de classe	Descrição
A	Retorna e define o parâmetro de início do intervalo crescente.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
B	Retorna e define o primeiro parâmetro de núcleo do conjunto difuso.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembership Function func3(2,7);
//--- Create wrappers for membership functions
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna o parâmetro de início do intervalo crescente.

```
double  A()
```

Valor de retorno

Parâmetro de início do intervalo crescente.

A (Método Set)

Define o parâmetro de início do intervalo crescente.

```
void  A(
   const double  a      // parâmetro de início do intervalo crescente
   )
```

Parâmetros

a

[in] Parâmetro de início do intervalo crescente.

B (Método Get)

Retorna o primeiro parâmetro de núcleo do conjunto difuso.

```
double  B()
```

Valor de retorno

Primeiro parâmetro de núcleo do conjunto difuso.

B (Método Set)

Define o primeiro parâmetro de núcleo do conjunto difuso.

```
void  B(
   const double  b      // primeiro parâmetro de núcleo do conjunto difuso
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

b

[in] Primeiro parâmetro de núcleo do conjunto difuso.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

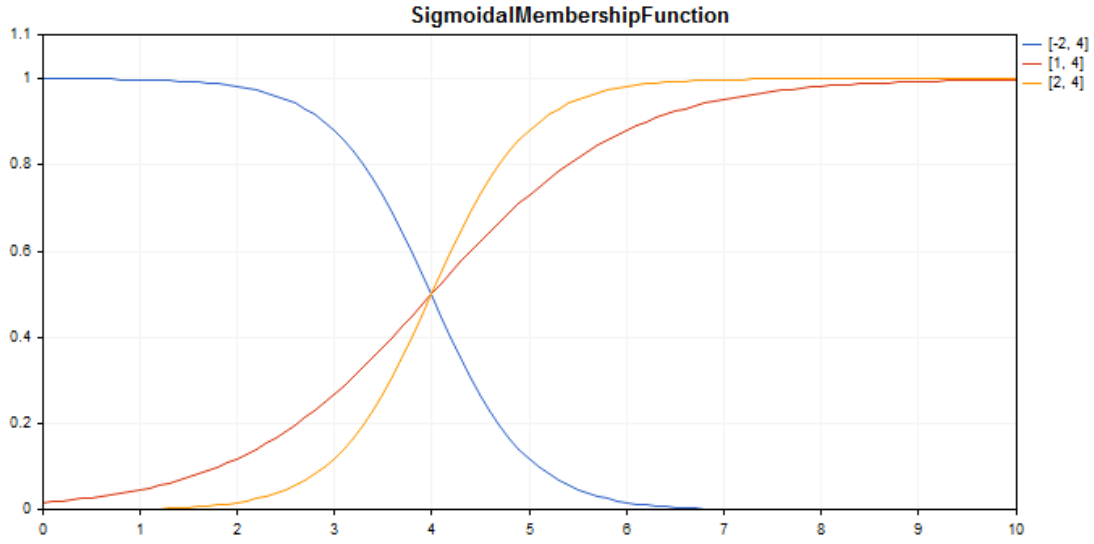
© 2000-2024, MetaQuotes Ltd.

CSigmoidalMembershipFunction

Classe para implementar uma função de associação sigmóide com parâmetros A e C.

Descrição

A fórmula sigmóide usa-se para definir funções de associação monótonas: Você pode usá-la para criar uma função de associação com valores iguais a 1, começando com um determinado valor do argumento. Estas funções são adequados, se é necessário definir os termos linguísticos de tipo "curto" ou "longo".



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CSigmoidalMembershipFunction

Métodos de classe

Método de classe	Descrição
A	Retorna e define o coeficiente de inclinação da função de associação.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
C	Retorna o parâmetro de coordenada da inflexão da função de associação.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- Create wrappers for membership functions
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembership Function2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna o coeficiente de inclinação da função de associação.

```
double  A()
```

Valor de retorno

Coeficiente de inclinação da função de associação.

A (Método Set)

Define o coeficiente de inclinação da função de associação.

```
void  A(
   const double  a      // coeficiente de inclinação da função de associação
   )
```

Parâmetros

a

[in] Coeficiente de inclinação da função de associação.

C (Método Get)

Retorna o parâmetro de coordenada da inflexão da função de associação.

```
double  C()
```

Valor de retorno

Coordenada de inflexão da função de associação.

C (Método Set)

Define a coordenada de inflexão da função de associação.

```
void  C(
   const double  c      // coordenada de inflexão da função de associação
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

c

[in] Coordenada de inflexão da função de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

© 2000-2024, MetaQuotes Ltd.

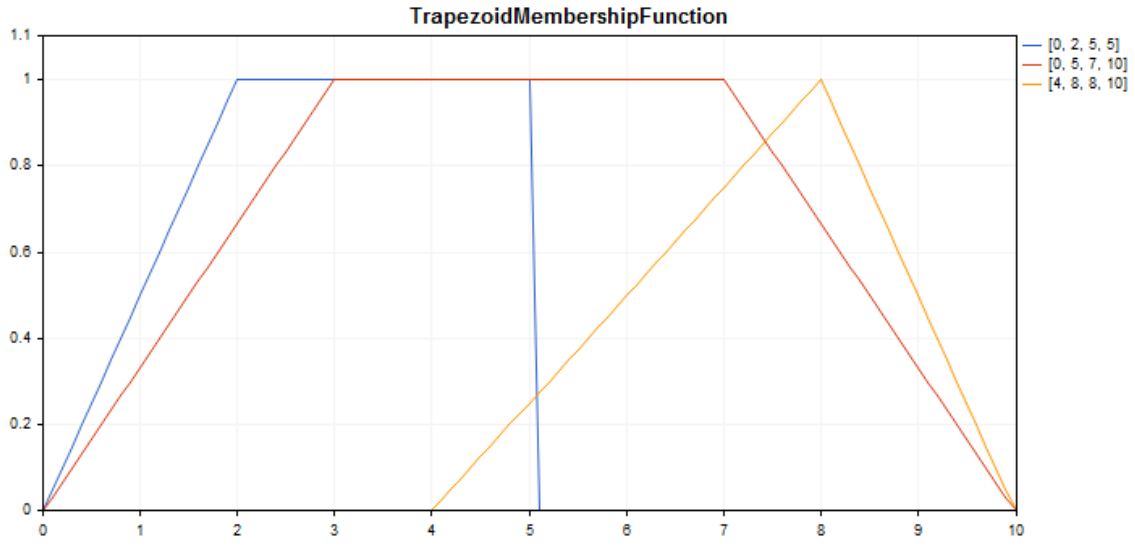
CTrapezoidMembershipFunction

Classe para implementar uma função de associação trapezoidal com parâmetros X1, X2, X3 e X4.

Descrição

A função é formada utilizando uma aproximação linear por partes. Trata-se de uma generalização da função triangular, ela permite definir o núcleo do conjunto difuso sob a forma de intervalo. Tal função de associação permite a fácil interpretação da avaliação otimista/pessimista.

Ela é utilizada para definir as funções de associação assimétricas de variáveis cujos valores mais possíveis são tomados num determinado intervalo.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CTrapezoidMembershipFunction

Métodos de classe

Método de classe	Descrição
X1	Retorna e define do primeiro ponto na abcissa.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
X2	Retorna e define o valor do segundo ponto na abcissa.
X3	Retorna e define o valor do terceiro ponto na abcissa.
X4	Retorna e define o valor do quarto ponto na abcissa.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- Create wrappers for membership functions
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembership Function3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5, 5
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7, 1
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8, 1
//--- sets the X-axis properties
```

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Método Get)

Retorna o valor do primeiro ponto na abcissa.

```
double  X1()
```

Valor de retorno

Retorna o valor do primeiro ponto na abcissa.

X1 (Método Set)

Define o valor do primeiro ponto na abcissa.

```
void  X1(
   const double  x1      // valor do primeiro ponto na abcissa
   )
```

Parâmetros

x1

[in] Retorna o valor do primeiro ponto na abcissa.

X2 (Método Get)

Retorna o valor do segundo ponto na abcissa.

```
double  X2()
```

Valor de retorno

Valor do segundo ponto na abcissa.

X2 (Método Set)

Define o valor do segundo ponto na abcissa.

```
void  X2(
   const double  x2      // valor do segundo ponto na abcissa
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

x2

[in] Valor do segundo ponto na abcissa.

X3 (Método Get)

Retorna o valor do terceiro ponto na abcissa.

```
double  X3()
```

Valor de retorno

Valor do terceiro ponto na abcissa.

X3 (Método Set)

Define o valor do terceiro ponto na abcissa.

```
void  X3(
   const double  x3      // valor do terceiro ponto na abcissa
   )
```

Parâmetros

x3

[in] Valor do terceiro ponto na abcissa.

X4 (Método Get)

Retorna o valor do quarto ponto na abcissa.

```
double  X4()
```

Valor de retorno

Valor do quarto ponto na abcissa.

X4 (Método Set)

Define o valor do quarto ponto na abcissa.

```
void  X4(
   const double  x4      // valor do quarto ponto na abcissa
   )
```

Parâmetros

x4

[in] Valor do quarto ponto na abcissa.

© 2000-2024, MetaQuotes Ltd.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

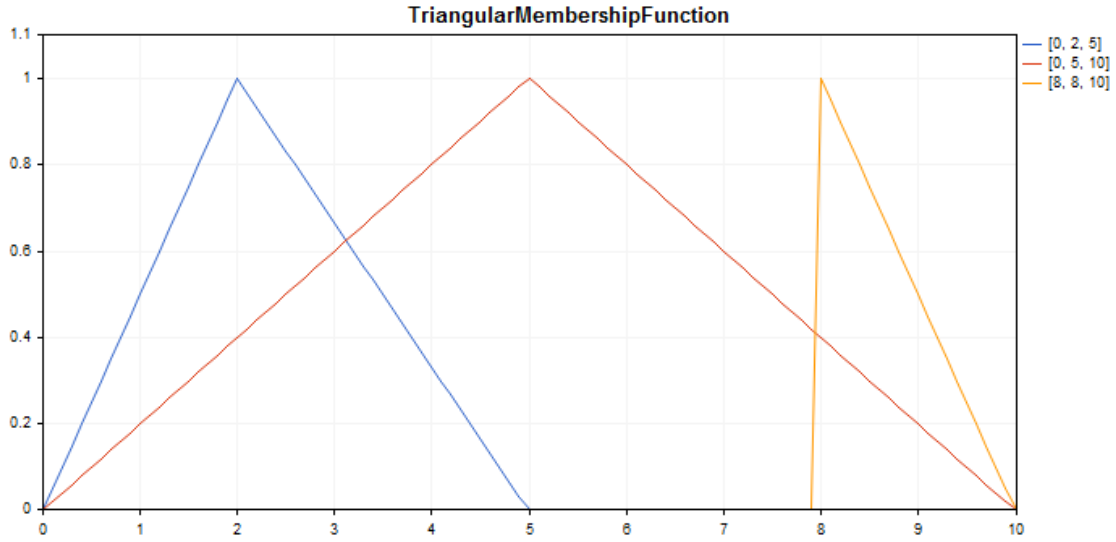
© 2000-2024, MetaQuotes Ltd.

CTriangularMembershipFunction

Classe para implementar uma função de associação triangular com parâmetros X1, X2, X3.

Descrição

A função defina a função de associação na forma de um triângulo. Trata-se da simples e frequentemente utilizada função de associação.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CTriangularMembershipFunction

Métodos de classe

Método de classe	Descrição
X1	Retorna o valor do primeiro ponto na abcissa.
X2	Retorna o valor do segundo ponto na abcissa.
X3	Retorna o valor do terceiro ponto na abcissa.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
ToNormalMF	Converte a função de associação triangular numa função de Gauss.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- Create wrappers for membership functions
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5]"
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 10]
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8, 10]
```

```
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Método Get)

Retorna o valor do primeiro ponto na abcissa.

```
double  X1()
```

Valor de retorno

Retorna o valor do primeiro ponto na abcissa.

X1 (Método Set)

Define o valor do primeiro ponto na abcissa.

```
void  X1(
   const double  x1      // valor do primeiro ponto na abcissa
   )
```

Parâmetros

x1

[in] Retorna o valor do primeiro ponto na abcissa.

X2 (Método Get)

Retorna o valor do segundo ponto na abcissa.

```
double  X2()
```

Valor de retorno

Valor do segundo ponto na abcissa.

X2 (Método Set)

Define o valor do segundo ponto na abcissa.

```
void  X2(
```

© 2000-2024, MetaQuotes Ltd.

```
   const double  x2      // valor do segundo ponto na abcissa
   )
```

Parâmetros

x2

[in] Valor do segundo ponto na abcissa.

X3 (Método Get)

Retorna o valor do terceiro ponto na abcissa.

```
double  X3()
```

Valor de retorno

Valor do terceiro ponto na abcissa.

X3 (Método Set)

Define o valor do terceiro ponto na abcissa.

```
void  X3(
   const double  x3      // valor do terceiro ponto na abcissa
   )
```

Parâmetros

x3

[in] Valor do terceiro ponto na abcissa.

ToNormalMF

Converte a função de associação triangular numa função de Gauss.

```
CNormalMembershipFunction*  ToNormalMF()
```

Valor de retorno

Ponteiro para a Funções de Gauss de associação.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

© 2000-2024, MetaQuotes Ltd.

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

© 2000-2024, MetaQuotes Ltd.

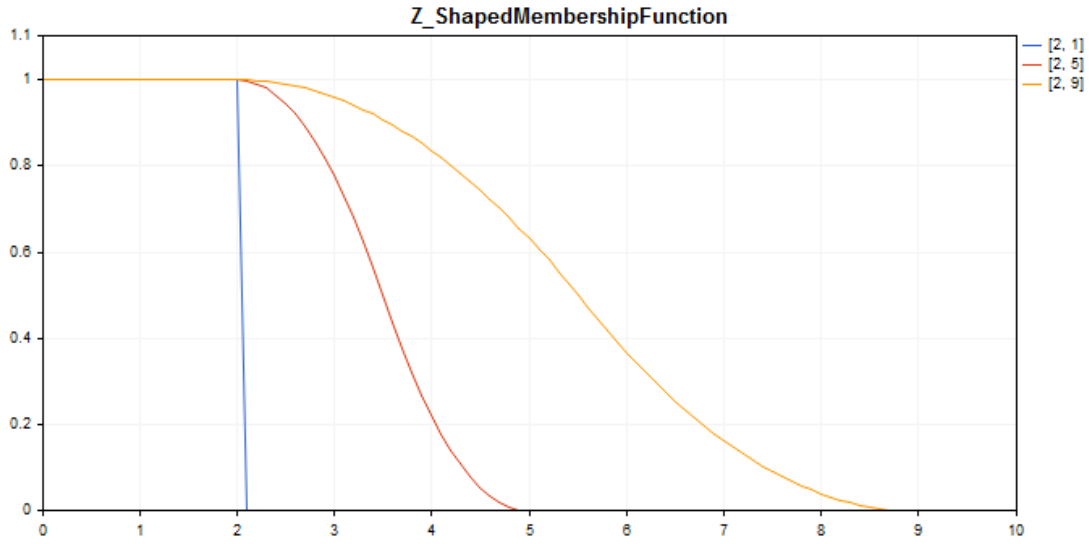
CZ_ShapedMembershipFunction

Classe para implementar funções de associação em forma de Z com parâmetros A e B.

Descrição

A função define a função de associação em forma de Z de dois parâmetros. Trata-se de uma função de associação não-crescente que aceita valores 1a 0. Os parâmetros da função definem o intervalo dentro do qual a função não linear diminui de 0 a 1.

Usando a função podem ser representados conjuntos difusos de tipo "muito baixo". Ou seja, ele define a função de associação não-crescente com saturação.



Exemplo de código para plotagem do gráfico mostrado abaixo.

Declaração

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject
  IMembershipFunction
    CZ_ShapedMembershipFunction

Métodos de classe

Método de classe	Descrição
A	Retorna e define o parâmetro de início do intervalo decrescente.

© 2000-2024, MetaQuotes Ltd.

Método de classe	Descrição
B	Retorna e define o parâmetro de fim do intervalo decrescente.
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Exemplo

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Create membership functions
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembership Function func3(2,9);
//--- Create wrappers for membership functions
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- create curve
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- sets the X-axis properties
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- sets the Y-axis properties
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Método Get)

Retorna o parâmetro de início do intervalo decrescente.

```
double  A()
```

Valor de retorno

Parâmetro de início do intervalo decrescente.

A (Método Set)

Define o parâmetro de início do intervalo decrescente.

```
void  A(
   const double  a      // parâmetro de início do intervalo decrescente
   )
```

Parâmetros

a

[in] Parâmetro de início do intervalo decrescente.

B (Método Get)

Retorna o parâmetro de fim do intervalo decrescente.

```
double  B()
```

Valor de retorno

Parâmetro de fim do intervalo decrescente.

B (Método Set)

Define o parâmetro de fim do intervalo decrescente.

```
void  B(
   const double  b      // parâmetro de fim do intervalo decrescente
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

b

[in] Parâmetro do fim do intervalo decrescente.

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

© 2000-2024, MetaQuotes Ltd.

IMembershipFunction

Classe base para todas as classes da função de associação.

Declaração

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Cabeçalho

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Hierarquia de herança

CObject

IMembershipFunction

Descendentes diretos

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

Métodos de classe

Método de classe	Descrição
GetValue	Calcula o valor da função de associação para o argumento especificado.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Calcula o valor da função de associação para o argumento especificado.

```
double  GetValue(
   const  x      // argumento de associação
   )
```

Parâmetros

x

[in] Argumento da função de associação.

Valor de retorno

Valor da função de associação.

© 2000-2024, MetaQuotes Ltd.

Regras para sistemas difusos

Chama-se **sistema difuso** (sistema de inferência difusa) à obtenção de uma conclusão na forma de conjunto difuso correspondente aos valores atuais das entradas com uso de um conjunto de **regras difusas** e operações difusas.

As **regras difusas** definem a relação entre as entradas e saídas do objeto pesquisado. O número de regras não está limitado, no sistema. Formato generalizado de regras difusas:

se há *condição (envio) de regra*, **então** há *conclusão quanto à regra*.

Se a *condição da regra* caracteriza o estado atual do objeto. A *condição da regra* descreve como a condição afetará o objeto.

Classe de regras para sistemas difusos	Descrição
CMamdaniFuzzyRule	Classe para implementar a regra de lógica difusa para algoritmo Mamdani
CSugenoFuzzyRule	Classe para implementar a regra de lógica difusa para algoritmo Sugeno
CSingleCondition	A classe define a condição difusa expressada pelo par "variável difusa - Termo difuso."
CConditions	A classe define o conjunto de condições difusas do operador.
CGenericFuzzyRule	Classe base para implementar ambos os tipos de regras difusas.

© 2000-2024, MetaQuotes Ltd.

CMamdaniFuzzyRule

A inferência difusa Mamdani faz parte de um dos dois principais tipos de sistemas difusos. No sistema Mamdani, os valores da variável de saída são definidos por termos difusos.

Descrição

A regra difusa para o algoritmo Mamdani pode ser descrita pela seguinte expressão:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

Onde:

- X = (X1, X2, X3 ... Xn) — veto de variáveis de entrada;
- Y — variável de saída;
- a = (a1, a2, a3 ... an) — vetor de valores de variáveis de entrada;
- d — valor da variável de saída;
- W — peso da regra.

Declaração

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hierarquia de herança

CObject
  IParsableRule
    CGenericFuzzyRule
      CMamdaniFuzzyRule

Métodos de classe

Método de classe	Descrição
Conclusion	Retorna e define a conclusão da regra difusa Mamdani.
Weight	Retorna e define o peso da regra difusa Mamdani.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Método Get)

© 2000-2024, MetaQuotes Ltd.

Retorna a conclusão da regra difusa Mamdani.

```
CSingleConditon*  Conclusion()
```

Valor de retorno

Conclusão da regra difusa Mamdani.

Conclusion (Método Set)

Define a conclusão da regra difusa Mamdani.

```
void  Conclusion(
   CSingleConditon*  value      // conclusão da regra difusa Mamdani
   )
```

Parâmetros

value

[in] Conclusão da regra difusa Mamdani.

Weight (Método Get)

Retorna o peso da regra difusa Mamdani.

```
double  Weight()
```

Valor de retorno

Peso da regra difusa Mamdani.

Weight (Método Set)

Define o peso da regra difusa Mamdani.

```
void  Weight(
   const double  value      // peso da regra difusa Mamdani
   )
```

Parâmetros

value

[in] Peso da regra difusa Mamdani.

© 2000-2024, MetaQuotes Ltd.

CSugenoFuzzyRule

A inferência difusa Sugeno faz parte de um dos dois principais tipos de sistemas difusos. O valor da variável de saída - nela - é definida como uma combinação linear de variáveis de entrada.

Descrição

Ao contrário da regra Mamdani, consiste no fato do valor da variável de saída não ser definido por um termo difuso, mas sim pela função linear a partir das entradas. A regra difusa para o algoritmo Sugeno pode ser descrita pela seguinte expressão:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

Onde:

- X = (X1, X2, X3 ... Xn) — veto de variáveis de entrada;
- Y — variável de saída;
- a = (a1, a2, a3 ... an) — vetor de valores de variáveis de entrada;
- b = (b1, b2, b3 ... bn) — coeficiente para o valor de saída, ao ter um termo livre na função linear
- W — peso da regra.

Declaração

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hierarquia de herança

CObject
  IParsableRule
    CGenericFuzzyRule
      CSugenoFuzzyRule

Métodos de classe

Método de classe	Descrição
Conclusion	Retorna e define a conclusão da regra difusa Sugeno.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Método Get)

© 2000-2024, MetaQuotes Ltd.

Retorna a conclusão da regra difusa Sugeno.

```
CSingleConditon*  Conclusion()
```

Valor de retorno

Conclusão da regra difusa Sugeno.

Conclusion (Método Set)

Define a conclusão da regra difusa Sugeno.

```
void  Conclusion(
   CSingleCondition*  value      // conclusão da regra difusa Sugeno
   )
```

Parâmetros

value

[in] Conclusão da regra difusa Sugeno.

© 2000-2024, MetaQuotes Ltd.

CSingleCondition

A classe define a condição difusa expressada pelo par "variável difusa - Termo difuso."

Descrição

A cada condição difusa de uma variável corresponde a um termo. A condição difusa pode ser descrita pela seguinte expressão: *X is a*,

Onde:

- X — variável difusa;
- a — valor da variável difusa (termo difuso).

Declaração

```
class CSingleCondition : public ICondition
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hierarquia de herança

CObject
  ICondition
    CSingleCondition

Descendentes diretos

CFuzzyCondition

Métodos de classe

Método de classe	Descrição
Not	Retorna e define o sinalizador que indica se é necessário aplicar uma negação a esta condição.
Term	Retorna e define o termo difuso para esta condição.
Var	Retorna e define o termo difuso para esta condição.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (Método Get)

Retorna o sinalizador que indica se é necessário aplicar uma negação a esta condição.

```
bool Not()
```

Valor de retorno

© 2000-2024, MetaQuotes Ltd.

Valor do sinalizador.

Not (Método Set)

Define o sinalizador que indica se é necessário aplicar uma negação a esta condição..

```
void  Not(
   bool  not      // valor do sinalizador
   )
```

Parâmetros

not

[in] Valor do sinalizador.

Term (Método Get)

Retorna o termo difuso para esta condição.

```
INamedValue* Term()
```

Valor de retorno

Termo difuso para esta condição.

Term (Método Set)

Define o termo difuso para esta condição.

```
void  Term(
   INamedValue*&  value      // termo difuso para esta condição
   )
```

Parâmetros

value

[in] Termo difuso para esta condição.

Var (Método Get)

Retorna a variável difusa para esta condição.

```
INamedVariable* Var()
```

Valor de retorno

Variável difusa para esta condição.

Var (Método Set)

Define a variável difusa para esta condição.

```
void  Var(
   INamedVariable*&  value      // variável difusa para esta condição
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

value

[in] variável difusa.

© 2000-2024, MetaQuotes Ltd.

CConditions

A classe define o conjunto de condições difusas do operador.

Descrição

O conjunto de condições difusas associadas ao operador pode ser descrito pela seguinte expressão::

$$(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)$$

Onde:

- X = (X1, X2, X3 ... Xn) — veto de variáveis de entrada;
- a = (a1, a2, a3 ... an) — vetor de valores de variáveis de entrada.

Neste exemplo, é usado o operador *and* (e). Nesta classe, também está disponível o operador *or* (ou).

Declaração

```
class CConditions : public ICondition
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hierarquia de herança

CObject
  ICondition
    CConditions

Métodos de classe

Método de classe	Descrição
ConditionsList	Retorna a lista de todas as condições
Not	Retorna o sinalizador que indica se é necessário aplicar uma negação a estas condições
Op	Retorna e define o tipo de operador de implicação de condições.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Retorna a lista de todas as condições.

```
CList*  ConditionsList()
```

© 2000-2024, MetaQuotes Ltd.

Valor de retorno

Lista de todas as condições.

Not (Método Get)

Retorna o sinalizador que indica se é necessário aplicar uma negação a estas condições.

```
bool Not()
```

Valor de retorno

Valor do sinalizador.

Not (Método Set)

Define o sinalizador que indica se é necessário aplicar uma negação a estas condições.

```
void  Not(
   bool  not      // valor do sinalizador
   )
```

Parâmetros

not

[in] Valor do sinalizador.

Op (Método Get)

Retorna o tipo de operador de implicação de condições. Estão disponíveis dois tipos de operadores: *and* (e) e *or* (ou).

```
OperatorType Op()
```

Valor de retorno

Tipo de operador de implicação de condições.

Op (Método Set)

Define o operador de implicação de condições. Estão disponíveis dois tipos de operadores: *and* (e) e *or* (ou).

```
void  Op(
   OperatorType  op     // tipo de operador de implicação de condições
   )
```

Parâmetros

op

[in] Tipo de operador de implicação de condições.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

CGenericFuzzyRule

Classe base para ambos os tipos de regras difusas.

Declaração

```
class CGenericFuzzyRule : public IParsableRule
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Hierarquia de herança

CObject
IParsableRule
CGenericFuzzyRule

Descendentes diretos

CMamdaniFuzzyRule, CSugenoFuzzyRule

Métodos de classe

Método de classe	Descrição
Conclusion	Retorna e define a conclusão da regra difusa
Condition	Retorna e define a condição (conjunto de condições) "if" para a regra difusa
CreateCondition	Cria a condição para a regra difusa segundo os parâmetros definidos

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (Método Get)

Retorna a conclusão da regra difusa.

```
CSingleConditon*  Conclusion()
```

Valor de retorno

Conclusão da regra difusa.

Conclusion (Método Set)

Define a conclusão da regra difusa.

```
virtual void  Conclusion(
   CSingleConditon*  value      // conclusão da regra difusa
```

© 2000-2024, MetaQuotes Ltd.

```
   )
```

Parâmetros

value

[in] Conclusão da regra difusa.

Condition (Método Get)

Retorna a condição (conjunto de condições) "if" para a regra difusa

```
CConditons*  Condition()
```

Valor de retorno

Condição difusa (conjunto de condições).

Condition (Método Set)

Define a condição (conjunto de condições) "if" para a regra difusa

```
void  Condition(
   CConditons*  value      // condição (conjunto de condições) "if" para a regra difus
   )
```

Parâmetros

value

[in] Condição difusa (conjunto de condições).

CreateCondition

Cria a condição para a regra difusa segundo os parâmetros definidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variável difusa
   CFuzzyTerm*      term,      // termo difuso
   )
```

Parâmetros

var

[in] Variável difusa.

term

[in] Termo difuso.

Valor de retorno

Estado da regra difusa.

CreateCondition

© 2000-2024, MetaQuotes Ltd.

Cria a condição para a regra difusa segundo os parâmetros definidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // variável difusa
   CFuzzyTerm*      term,      // termo difuso
   bool             not,       // sinalizador que indica se é necessário aplicar uma ne
   )
```

Parâmetros

var

[in] Variável difusa.

term

[in] Termo difuso.

not

[in] Sinalizador que indica se é necessário aplicar uma negação a esta condição.

Valor de retorno

Estado da regra difusa.

CreateCondition

Cria a condição para a regra difusa segundo os parâmetros definidos.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,       // variável difusa
   CFuzzyTerm*     term,      // termo difuso
   bool            not,       // sinalizador que indica se é necessário aplicar uma neg
   HedgeType       hedge      // tipo de implicação de condição
   )
```

Parâmetros

var

[in] Variável difusa.

term

[in] Termo difuso.

not

[in] Sinalizador que indica se é necessário aplicar uma negação a esta condição.

hedge

[in] Tipo de implicação de condição.

Valor de retorno

Estado da regra difusa.

© 2000-2024, MetaQuotes Ltd.

Variáveis para sistemas difusos

Nos sistemas difusos são aplicadas **variáveis (linguísticas) difusas**. Tratam-se de variáveis cujos valores podem ser palavras ou frases de uma língua natural ou artificial.

A variáveis linguísticas constituem os conjuntos difusos. Ao definir conjuntos difusos, a natureza e quantidade das variáveis difusas variam para cada tarefa individual.

Classe	Descrição
CFuzzyVariable	Classe com cuja ajuda são criadas variáveis difusas de forma geral.
CSugenoVariable	Classe com cuja ajuda são criadas variáveis difusas de forma geral.

© 2000-2024, MetaQuotes Ltd.

CFuzzyVariable

Classe com cuja ajuda são criadas variáveis difusas de forma geral.

Descrição

Neste caso, a variável difusa é definida pelos seguintes parâmetros:

- valor máximo de variável;
- valor mínimo de variável;
- nome da variável difusa;
- termo-conjunto (conjunto de todos os valores possíveis que podem ser aceites pela variável linguística).

Declaração

```
class CFuzzyVariable : public CNamedVariableImpl
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Hierarquia de herança

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CFuzzyVariable

Métodos de classe

Método de classe	Descrição
AddTerm	Adiciona um termo difuso à variável difusa.
GetTermByName	Obtém o termo difuso de acordo com o nome especificado.
Max	Retorna e define o valor máximo para a variável difusa.
Min	Retorna e define o valor mínimo para a variável difusa.
Terms	Retorna e define a lista de termos difusos para esta variável difusa.
Values	Retorna e define a lista de termos difusos para esta variável difusa.

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CNamedVariableImpl

Name, Name

AddTerm

Adiciona um termo difuso à variável difusa.

```
void  AddTerm(
   CFuzzyTerm*&  term      // termo difuso
   )
```

Parâmetros

term

[in] Termo difuso.

GetTermByName

Obtém o termo difuso de acordo com o nome especificado.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // nome do termo difuso
   )
```

Parâmetros

name

[in] Nome do termo difuso.

Valor de retorno

Termo difuso com o nome especificado.

Max (Método Get)

Retorna o valor máximo para a variável difusa.

```
double  Max()
```

Valor de retorno

Retorna o valor máximo para a variável difusa.

Max (Método Set)

Define o valor máximo para a variável difusa.

```
void  Max(
   const double  max      // valor máximo para a variável difusa
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

max

[in] Valor máximo para a variável difusa.

Min (Método Get)

Retorna o valor mínimo para a variável difusa.

```
double  Min()
```

Valor de retorno

Retorna o valor mínimo para a variável difusa.

Max (Método Set)

Define o valor mínimo para a variável difusa.

```
void  Min(
   const double  min      // valor mínimo para a variável difusa
   )
```

Parâmetros

min

[in] Valor mínimo para a variável difusa.

Terms (Método Get)

Retorna a lista de termos difusos para essa variável difusa.

```
CList*  Terms()
```

Valor de retorno

Lista de termos difusos para esta variável difusa.

Terms (Método Set)

Define a lista de termos difusos para esta variável difusa.

```
void  Terms(
   CList*&  terms      // lista de termos difusos para esta variável
   )
```

Parâmetros

terms

[in] lista de termos difusos para esta variável difusa.

© 2000-2024, MetaQuotes Ltd.

Values

Retorna a lista de termos difusos para essa variável difusa.

```
CList*  Values()
```

Valor de retorno

Lista de termos difusos para essa variável.

© 2000-2024, MetaQuotes Ltd.

CSugenoVariable

Classe com cuja ajuda são criadas variáveis difusas de forma geral.

Descrição

A variável difusa do tipo Sugeno é diferente da variável linguística de tipo geral no fato de que não está definida por um termo-conjunto, mas sim por um conjunto de funções lineares.

Declaração

```
class CSugenoVariable : public CNamedVariableImpl
```

Cabeçalho

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Hierarquia de herança

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CSugenoVariable

Métodos de classe

Método de classe	Descrição
Functions	Retorna a lista de funções lineares da variável difusa Sugeno.
GetFuncByName	Retorna função linear para o nome especificado.
Values	Retorna a lista de funções lineares da variável difusa Sugeno.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CNamedVariableImpl

Name, Name

Functions

Retorna a lista de funções lineares da variável difusa Sugeno.

```
CList*  Functions()
```

Valor de retorno

© 2000-2024, MetaQuotes Ltd.

Lista de funções lineares.

GetFuncByName

Retorna função linear para o nome especificado.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // nome da função linear
   )
```

Parâmetros

name

[in] Nome da função linear.

Valor de retorno

Função linear com o nome especificado.

Values

Retorna a lista de funções lineares da variável difusa Sugeno.

```
CList*  Values()
```

Valor de retorno

Lista de funções lineares da variável difusa Sugeno.

© 2000-2024, MetaQuotes Ltd.

CFuzzyTerm (termos difusos)

Classe para implementação de termos difusos.

Descrição

Um **termo (term)** é qualquer elemento de um conjunto de termos. O termo é determinado por dois componentes:

- o nome do termo difuso;
- a funções de associação.

Declaração

```
class CFuzzyTerm : public CNamedValueImpl
```

Cabeçalho

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Hierarquia de herança

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

Métodos de classe

Método de classe	Descrição
MembershipFunction	Retorna a função de associação para esse termo difuso.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CNamedValueImpl

Name, Name

© 2000-2024, MetaQuotes Ltd.

MembershipFunction

Retorna a função de associação para esse termo difuso.

```
IMembershipFunction*  MembershipFunction()
```

Valor de retorno

Função de associação.

© 2000-2024, MetaQuotes Ltd.

Sistemas difusos

Um **sistema difuso** (ou *modelo difuso*) é um modelo matemático, cuja base de cálculo é o fundamento da lógica difusa. Recorre-se à plotagem de tais modelos em caso de o objeto de estudo ter uma formalização muito fraca e sua descrição matemática ser muito complicada ou desconhecida.

O progresso na plotagem do modelo pode ser dividida em três fases principais:

1. Determinação dos parâmetros de entrada e de saída do modelo.
2. Construção de uma base de conhecimento.
3. Seleção de um dos métodos de inferência difusa (Mamdani ou Sugeno).

Os outros dois dependem da primeira fase, uma vez que ela define o funcionamento futuro do modelo.

A **base de conhecimentos** () abrange o conjunto de regras difusas do tipo: "se, então" que definem a relação entre as entradas e saídas do objeto pesquisado.

A **condição (Condition)** é a regra que carateriza o estado atual do objeto, enquanto a **conclusão (Conclusion)** trata de como a condição afeta o objeto.

As condições e conclusões de cada regra podem ser de dois tipos:

1. simples (ligação para Csinglcond) — nela está envolvida uma variável difusa;
2. composta (ligação para Cconditions) — participam várias variáveis difusas.

Cada regra no sistema possui **peso** — relevância de regra no modelo. Os fatores de ponderação são atribuídos à regra no intervalo [0, 1].

O sistema de inferência difuso é determinado dependendo da base de conhecimento criada para o modelo. Chama-se **inferência difusa** à obtenção de uma conclusão na forma de conjunto difuso correspondente aos valores atuais das entradas com uso de uma base difusa de conhecimentos e operações difusas. Há dos tipos básicos de inferência difusa, isto é, Mamdani ou Sugeno.

© 2000-2024, MetaQuotes Ltd.

Sistema Mamdani

No sistema Mamdani, os valores da variável de saída são definidos por termos difusos.

Descrição

A regra difusa para o algoritmo Mamdani pode ser descrita pela seguinte expressão:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

Onde:

- X = (X1, X2, X3 ... Xn) — veto de variáveis de entrada;
- Y — variável de saída;
- a = (a1, a2, a3 ... an) — vetor de valores de variáveis de entrada;
- d — valor da variável de saída;
- W — peso da regra.

Métodos de classe

Método de classe	Descrição
AggregationMethod	Define o tipo de condições de agregação
Calculate	Calcula a inferência difusa para este sistema
DefuzzificationMethod	Define o tipo de método de exclusão da difusão
EmptyRule	Cria a regra difusa Mamdani vazia na base do sistema atual
ImplicationMethod	Define o tipo de operador de implicação de condições
Output	Retorna a lista de variáveis difusas Mamdani de saída.
OutputByName	Retorna a variável difusa Mamdani de saída para o nome especificado.
ParseRule	Cria a regra difusa Mamdani na base da cadeia de caracteres atual.
Rules	Retorna a lista de regras difusas Mamdani.

Métodos herdados da classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Define o tipo de método de agregação de condições.

```
void  AggregationMethod(
   AggregationMethod  value      // tipo de método de agregação
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

value

[in] Tipo de método de agregação de condições.

Calculate

Calcula a inferência difusa para este sistema.

```
CList*  Calculate(
   CList*  inputValues      // parâmetros de entrada
   )
```

Parâmetros

inputValues

[in] Parâmetros de entrada para o cálculo.

Valor de retorno

Resultados dos cálculos.

DefuzzificationMethod

Define o tipo de método de exclusão da difusão.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // tipo de método de exclusão da difusão.
   )
```

Parâmetros

value

[in] Tipo de método de exclusão da difusão.

EmptyRule

Cria a regra difusa Mamdani vazia na base do sistema atual.

```
CMamdaniFuzzyRule*  EmptyRule()
```

Valor de retorno

Regra difusa Mamdani.

ImplicationMethod

Define o tipo de operador de implicação de condições.

```
void  ImplicationMethod(
   ImplicationMethod  value      // tipo de operador de implicação
```

© 2000-2024, MetaQuotes Ltd.

```
    )
```

Parâmetros

value

[in] Tipo de operador de implicação de condições.

Output

Retorna a lista de variáveis difusas Mamdani de saída.

```
CList*  Output()
```

Valor de retorno

Lista de variáveis difusas.

OutputByName

Retorna a variável difusa Mamdani de saída para o nome especificado.

```
CFuzzyVariable*  OutputByName(
   const string  name      // nome da variável difusa
   )
```

Parâmetros

name

[in] Nome da variável difusa.

Valor de retorno

Variável difusa Mamdani com o nome especificado.

ParseRule

Cria a regra difusa Mamdani na base da cadeia de caracteres atual.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // representação sob a forma de cadeia de caracteres da re
   )
```

Parâmetros

rule

[in] Representação de cadeia de caracteres da regra difusa Mamdani.

Valor de retorno

Regra difusa Mamdani.

Rules

© 2000-2024, MetaQuotes Ltd.

Retorna a lista de regras difusas Mamdani.

```
CList*  Rules()
```

Valor de retorno

Lista de regras difusas Mamdani

© 2000-2024, MetaQuotes Ltd.

Sistema Sugeno

O sistema lógico difuso Sugeno faz parte de um dos dois principais tipos de sistemas difusos. O valor da variável de saída - nela - é definida como uma combinação linear de variáveis de entrada.

Descrição

Ao contrário do sistema Mamdani, consiste no fato do valor da variável de saída não ser definido por um termo difuso, mas sim pela função linear a partir das entrada. A regra difusa para o algoritmo Sugeno pode ser descrita pela seguinte expressão:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

Onde:

- X = (X1, X2, X3 ... Xn) — veto de variáveis de entrada;
- Y — variável de saída;
- a = (a1, a2, a3 ... an) — vetor de valores de variáveis de entrada;
- b = (b1, b2, b3 ... bn) — coeficiente para o valor de saída, ao ter um termo livre na função linear
- W — peso da regra.

Métodos de classe

Método de classe	Descrição
Calculate	Calcula a inferência difusa para este sistema.
CreateSugenoFunction	Cria uma função linear Sugeno para o sistema atual.
EmptyRule	Cria uma regra difusa Sugeno vazia na base do sistema atual.
Output	Retorna a lista de variáveis difusas Sugeno de saída.
OutputByName	Retorna a variável difusa Sufeno de saída para o nome especificado.
ParseRule	Cria a regra difusa Sugeno na base da cadeia de caracteres definida.
Rules	Retorna a lista de regras difusas.

Métodos herdados da classe CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Calcula a inferência difusa para este sistema.

```
CList*  Calculate(
   CList*&  inputValues      // parâmetros de entrada
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

inputValues

[in] Parâmetros de entrada para o cálculo.

Valor de retorno

Resultados dos cálculos.

CreateSugenoFunction

Cria uma função linear Sugeno para o sistema atual.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nome da função
   const double& coeffs[]       // coeficiente da função
   )
```

Parâmetros

name

[in] Nome da função.

coeffs[]

[in] Coeficiente da função.

Valor de retorno

Função linear Sugeno.

Observação

O tamanho da matriz de coeficientes pode ser superior - por uma unidade - ao número de variáveis de entrada ou igual a ele. No primeiro caso, o membro livre da função linear Sugeno ao último coeficiente, enquanto, no segundo caso, será igual a zero.

CreateSugenoFunction

Cria uma função linear Sugeno para o sistema atual.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // nome da função
   CList*&       coeffs,        // a lista de pares da variável difusa é o coeficiente
   const double  constValue     // coeficiente ao ter o membro livre da função
   )
```

Parâmetros

name

[in] Nome da função.

coeffs[]

[in] Coeficiente da função.

© 2000-2024, MetaQuotes Ltd.

Valor de retorno

Função linear Sugeno.

EmptyRule

Cria uma regra difusa Sugeno vazia na base do sistema atual.

```
CSugenoFuzzyRule*  EmptyRule()
```

Valor de retorno

Regra difusa Sugeno.

Output

Retorna a lista de variáveis difusas Sugeno de saída.

```
CList*  Output()
```

Valor de retorno

Lista de variáveis difusas.

OutputByName

Retorna a variável difusa Sufeno de saída para o nome especificado.

```
CSugenoVariable*  OutputByName(
   const string  name      // nome da variável difusa
   )
```

Parâmetros

name

Nome da variável difusa.

Valor de retorno

Variável difusa Sugeno com o nome especificado.

ParseRule

Cria a regra difusa Sugeno na base da cadeia de caracteres definida.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // representação sob a forma de cadeia de caracteres da reg
   )
```

Parâmetros

rule

[in] Representação de cadeia de caracteres da regra difusa Sugeno.

© 2000-2024, MetaQuotes Ltd.

Valor de retorno

Regra difusa Sugeno.

Rules

Retorna a lista de regras difusas.

```
CList*  Rules()
```

Valor de retorno

Lista de regras difusas.

© 2000-2024, MetaQuotes Ltd.

Classe para trabalhar com programas OpenCL

A classe COpenCL é um wrapper (pano de embrulho) para facilitar o trabalho com as funções OpenCL. O uso da GPU, em alguns casos, pode aumentar significativamente a velocidade de computação.

Exemplos de utilização de classe para cálculos nos números float e double podem ser encontrados na pasta MQL5\Scripts\Examples\OpenCL\ nos respectivos sub-diretórios, os códigos-fontes dos programas OpenCL residem nos sub-diretórios MQL5\Scripts\Examples\OpenCL\Double\Kernels e MQL5\Scripts\Examples\OpenCL\Float\Kernels .

- MatrixMult.mq5 - exemplo de multiplicação de matrizes usando as memórias global e local
- BitonicSort.mq5 - exemplo de classificação paralela de elementos de matriz na GPU
- FFT.mq5 - exemplo de cálculo de transformada rápida de Fourier
- Wavelet.mq5 - exemplo de cálculo da transformação de dados por ondulação usando a transformação por ondulação Morlet.

É recomendado escrever o código-fonte OpenCL em arquivos CL separados, que podem então ser ligados ao programa MQL5 usando variáveis de recursos.

Declaração

```
class COpenCL
```

Cabeçalho

```
#include <OpenCL\COpenCL.mqh>
```

Métodos de classe

Nome	Descrição
BufferCreate	Cria o buffer OpenCL segundo o índice especificado
BufferFree	Remove o buffer segundo o índice especificado
BufferFromArray	Cria o buffer segundo o índice especificado a partir da matriz de valores
BufferRead	Lê - na matriz - o buffer OpenCL segundo o índice especificado
BufferWrite	Escreve a matriz de dados no buffer segundo o índice especificado
Execute	Executa o programa OpenCL segundo o índice especificado
GetContext	Retorna o identificador de contexto OpenCL
GetKernel	Retorna o identificador de objeto kernel segundo o índice especificado
GetKernelName	Retorna o nome do objeto kernel segundo o índice especificado
GetProgram	Retorna o identificador de programa OpenCL
Initialize	Inicializa o programa OpenCL

© 2000-2024, MetaQuotes Ltd.

Nome	Descrição
KernelCreate	Cria o ponto de entrada para o programa OpenCL segundo o índice especificado
KernelFree	Exclui a função que inicia OpenCL segundo o índice especificado
SetArgument	Define o parâmetro para a função OpenCL segundo o índice especificado
SetArgumentBuffer	Define o buffer OpenCL como um parâmetro da função OpenCL segundo o índice especificado
SetArgumentLocalMemory	Define o parâmetro na memória local para a função OpenCL segundo o índice especificado
SetBuffersCount	Define o número de buffers
SetKernelsCount	Define o número de objetos kernel
Shutdown	Descarrega o programa OpenCL
SupportDouble	Confere a presença de suporte - no dispositivo - de tipos de dados reais

© 2000-2024, MetaQuotes Ltd.

BufferCreate

Cria o buffer OpenCL segundo o índice especificado.

```
bool  BufferCreate(
   const int   buffer_index,      // índice de buffer
   const uint  size_in_bytes,     // tamanho de buffer em bytes
   const uint  flags              // combinação de sinalizadores que definem as propri
   );
```

Parâmetros

buffer_index

[in] Índice de buffer.

size_in_bytes

[in] Tamanho do buffer em bytes.

flags

[in] Propriedades do buffer a ser definidas através da combinação de sinalizadores.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

BufferFree

Remove o buffer segundo o índice especificado.

```
bool  BufferFree(
   const int  buffer_index      // índice de buffer
   );
```

Parâmetros

buffer_index

[in] Índice de buffer.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

BufferFromArray

Cria o buffer segundo o índice especificado a partir da matriz de valores.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // índice de buffer
   T           &data[],               // matriz de valores
   const uint  data_array_offset,     // deslocamento - em bytes - na matriz de valore
   const uint  data_array_count,      // número de valores a partir da matriz para reg
   const uint  flags                  // // combinação de sinalizadores que definem as
   );
```

Parâmetros

buffer_index

[in] Índice de buffer.

&data[]

[in] Matriz de valores que devem ser registrados no buffer OpenCL.

data_array_offset

[in] Deslocamento na matriz de valores em bytes, a partir do qual começa o registro.

data_array_count

[in] Número de valores que é preciso registrar.

flags

[in] Propriedades do buffer a ser definidas através da combinação de sinalizadores.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

BufferRead

Lê - na matriz - o buffer OpenCL segundo o índice especificado.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // índice de buffer
   T           &data[],               // matriz de valores
   const uint  cl_buffer_offset,      // deslocamento - em bytes - na matriz de valore
   const uint  data_array_offset,     // deslocamento na matriz, nos elementos
   const uint  data_array_count       // número de valores a partir do buffer para le
   );
```

Parâmetros

buffer_index

[in] Índice de buffer.

&data[]

[in] Matriz para obter valores a partir do buffer OpenCL.

cl_buffer_offset

[in] Deslocamento - em bytes - no buffer OpenCL, a partir do qual começa a leitura.

data_array_offset

[in] Índice do primeiro elemento da matriz para registrar os valores do buffer OpenCL.

data_array_count

[in] Número de valores que precisam de ser considerados.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

BufferWrite

Registra a matriz de dados no buffer segundo o índice especificado.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // índice de buffer
   T           &data[],               // matriz de valores
   const uint  cl_buffer_offset,      // deslocamento - em bytes - na matriz de valore
   const uint  data_array_offset,     // deslocamento na matriz, nos elementos
   const uint  data_array_count       // número de valores a partir da matriz para re
   );
```

Parâmetros

buffer_index

[in] Índice de buffer.

&data[]

[in] Matriz de valores que devem ser registrados no buffer OpenCL.

cl_buffer_offset

[in] Deslocamento - em bytes - no buffer OpenCL, a partir do qual começa a leitura.

data_array_offset

[in] Índice do primeiro elemento da matriz, a partir do qual são tomados os valores para registro no buffer OpenCL.

data_array_count

[in] Número de valores que é preciso registrar.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Execute

Executa o programa OpenCL segundo o índice especificado.

```
bool  Execute(
   const int   kernel_index,           // índice kernel
   const int   work_dim,               // dimensão do espaço de tarefas
   const uint  &work_offset[],         // deslocamento inicial no espaço de tarefas
   const uint  &work_size[]            // número total de tarefas
   );
```

Executa o programa OpenCL segundo o índice especificado com o número de tarefas definido no grupo local.

```
bool  Execute(
   const int   kernel_index,           // índice kernel
   const int   work_dim,               // dimensão do espaço de tarefas
   const uint  &work_offset[],         // deslocamento inicial no espaço de tarefas
   const uint  &work_size[],           // número total de tarefas
   const uint  &local_work_size[]      // número de tarefas no grupo local
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

work_dim

[in] Dimensão do espaço de tarefas.

&work_offset[]

[in] Deslocamento inicial no espaço de tarefas.

&work_size[]

[in] Tamanho do sub-conjunto de tarefas.

&local_work_size[]

[in] Tamanho do sub-conjunto locar de tarefas no grupo.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

GetContext

Retorna o identificador de contexto OpenCL.

```
int  GetContext();
```

Valor de retorno

Identificador de contexto OpenCL.

© 2000-2024, MetaQuotes Ltd.

GetKernel

Retorna o identificador de objeto kernel segundo o índice especificado.

```
int  GetKernel(
   const int  kernel_index      // índice kernel
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

Valor de retorno

Identificador de objeto kernel.

© 2000-2024, MetaQuotes Ltd.

GetKernelName

Retorna o nome do objeto kernel segundo o índice especificado.

```
string  GetKernelName(
   const int  kernel_index      // índice kernel
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

Valor de retorno

Nome do objeto kernel.

© 2000-2024, MetaQuotes Ltd.

GetProgram

Retorna o identificador de programa OpenCL.

```
int  GetProgram();
```

Valor de retorno

Identificador de programa OpenCL.

© 2000-2024, MetaQuotes Ltd.

Initialize

Inicializa o programa OpenCL.

```
bool  Initialize(
   const string  program,           // identificador de programa OpenCL
   const bool    show_log=true      // realizar o registro no log
   );
```

Parâmetros

program

[in] Identificador de programa OpenCL.

show_log=true

[in] Ativar o registro de mensagens no diário.

Valor de retorno

Retorna true, se a inicialização for bem-sucedida. Caso contrário, retorna false.

© 2000-2024, MetaQuotes Ltd.

KernelCreate

Cria o ponto de entrada para o programa OpenCL segundo o índice especificado.

```
bool  KernelCreate(
   const int     kernel_index,     // índice kernel
   const string  kernel_name       // nome do kernel
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

kernel_name

[in] Nome do objeto kernel.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

KernelFree

Exclui a função que inicia OpenCL segundo o índice especificado.

```
bool  KernelFree(
   const int  kernel_index      // índice kernel
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetArgument

Define o parâmetro para a função OpenCL segundo o índice especificado.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,      // índice kernel
   const int  arg_index,         // índice do argumento da função
   T          value              // código-fonte
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

arg_index

[in] Índice do argumento da função.

value

[in] Valor do argumento da função.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetArgumentBuffer

Define o buffer OpenCL como um parâmetro da função OpenCL segundo o índice especificado.

```
bool  SetArgumentBuffer(
   const int  kernel_index,      // índice kernel
   const int  arg_index,         // índice do argumento da função
   const int  buffer_index       // índice de buffer
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

arg_index

[in] Índice do argumento da função.

buffer_index

[in] Índice de buffer.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetArgumentLocalMemory

Define o parâmetro na memória local para a função OpenCL segundo o índice especificado.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // índice kernel
   const int  arg_index,             // índice do argumento da função
   const int  local_memory_size      // tamanho da memória local
   );
```

Parâmetros

kernel_index

[in] Índice do objeto kernel.

arg_index

[in] Índice do argumento da função.

local_memory_size

[in] Tamanho da memória local.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetBuffersCount

Define o número de buffers.

```
bool  SetBuffersCount(
   const int  total_buffers      // número de buffers
   );
```

Parâmetros

total_buffers

[in] Número total de buffers.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetKernelsCount

Define o número de objetos kernel.

```
bool  SetKernelsCount(
   const int  total_kernels      // número de kernels
   );
```

Parâmetros

total_kernels

[in] Número total de kernels.

Valor de retorno

Se for bem-sucedido, retorna true, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Shutdown

Descarrega o programa OpenCL.

```
void  Shutdown();
```

Valor de retorno

Não há valores de retorno.

© 2000-2024, MetaQuotes Ltd.

SupportDouble

Confere a presença de suporte - no dispositivo - de tipos de dados reais.

```
bool  SupportDouble();
```

Valor de retorno

Retorna true, se o dispositivo oferece suporte a tipos de dados reais.

© 2000-2024, MetaQuotes Ltd.

Classe Base CObject

CObject é a classe base para a construção de uma biblioteca padrão MQL5.

Descrição

Classe CObject permite a todos os seus descendentes fazerem parte de uma lista interligada. Também identifica uma série de métodos virtuais para prosseguimento da execução em classes descendentes.

Declaração

```
class CObject
```

Título

```
#include <Object.mqh>
```

Hierarquia de herança

CObject

Descendentes diretos

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

Métodos de classe

Atributos	
Prev	Obtém o valor do item anterior
Prev	Define o valor do item anterior
Next	Obtém o valor do elemento posterior
Next	Define o próximo elemento
Métodos para comparar	
virtual Compare	Retorna o resultado da comparação com outro objeto
Entrada/saída	
virtual Save	Grava objeto para um arquivo
virtual Load	Lê o objeto a partir do arquivo
virtual Type	Retorna o tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Prev

Obtém um ponteiro para o item anterior da lista .

```
CObject*  Prev()
```

Valor do Retorno

Ponteiro para o item anterior da lista. Se o item é listado em primeiro lugar, então retorna NULL.

Exemplo

```
//--- example for CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- set interconnect
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- use prev object
   CObject *object=object_second.Prev();
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Prev

Define o ponteiro para o item anterior da lista.

```
void  Prev(
   CObject*  object      // Pointer to the previous list item
   )
```

Parâmetros

object

[in] New ponteiro valor para o item anterior da lista.

Exemplo

```
//--- example for CObject::Prev(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- set interconnect
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- use objects
   //--- ...
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Next

Obtém um ponteiro para o próximo elemento da lista.

```
CObject*  Next()
```

Valor do Retorno

Ponteiro para o próximo item na lista. Se for o último item na lista, retorna NULL.

Exemplo

```
//--- example for CObject::Next()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- set interconnect
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- use next object
   CObject *object=object_first.Next();
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Next

Define o ponteiro para o próximo elemento da lista.

```
void  Next(
   CObject*  object      // Pointer to the next list item
   )
```

Parâmetros

object

[in] Novo ponteiro de valor para o próximo item da lista.

Exemplo

```
//--- example for CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- set interconnect
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- use objects
   //--- ...
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Compare

Compara o item de dados na lista com dados sobre outro elemento da lista.

```
virtual int  Compare(
   const CObject*  node,      // Node to compare with
   const int       mode=0     // Compare mode
   ) const
```

Parâmetros

node

[in] Ponteiro para um item da lista a comparar

mode=0

[in] Comparação de Variante

Valor do Retorno

0 - no caso dos itens da lista serem iguais, -1 - se o item da lista é menor do que o item na lista a comparar (nó), 1 - se o item da lista de maior do que o item na lista a comparar (nó).

Observação

Método Compare () na classe CObject sempre retorna 0 e não executa qualquer ação. Se você quiser comparar dados de classe derivada, o método Compare (...) deve ser implementado. O parâmetro de modo deve ser usado quando implementação for comparação multivariada.

Exemplo

```
//--- example for CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Object create error");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Object create error");
      delete object_first;
      return;
     }
   //--- set interconnect
   object_first.Next(object_second);
```

© 2000-2024, MetaQuotes Ltd.

```
   object_second.Prev(object_first);
   //--- compare objects
   int result=object_first.Compare(object_second);
   //--- delete objects
   delete object_first;
   delete object_second;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva lista de elementos de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen(), binário

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Observação

Método Save (int) na classe CObject sempre retorna verdadeiro e não executa qualquer ação. Se você deseja salvar os dados derivados de classe no arquivo, o método Save (int) deve ser implementado.

Exemplo

```
//--- example for CObject::Save(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- set objects data
   //--- . . .
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      FileClose(file_handle);
    }
  delete object;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega item de dados na lista a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // handle to file
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen(), binário

Valor do Retorno

verdadeiro - se concluído com êxito, falso - se com erro.

Observação

Método Load (int) na classe CObject sempre retorna verdadeiro e não executa qualquer ação. Se você deseja carregar os dados derivados de classe a partir de um arquivo, o método Load (int) deve ser implementado.

Exemplo

```
//--- example for CObject::Load(int)
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- use object
  //--- . . .
  delete object;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo.

```
virtual int  Type() const
```

Valor do Retorno

Identificador do tipo (para CObject - 0).

Exemplo

```
//--- example for CObject::Type()
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get objects type
   int type=object.Type();
   //--- delete object
   delete object;
  }
```

© 2000-2024, MetaQuotes Ltd.

Estruturas de Dados

Esta seção contém os detalhes técnicos de trabalhar com várias estruturas de dados (arrays, listas vinculadas, etc.) e descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Usando as classes de estruturas de dados, será economizado tempo ao criar armazenamentos de dados personalizados de vários formatos (incluindo estruturas de dados compostos).

A Biblioteca Padrão MQL5 (em termos de conjuntos de dados) é colocada no diretório de trabalho do terminal, na pasta Include\Arrays.

Arrays de dados

Uso de classes de Arrays dinâmicos de dados economizará tempo ao criar armazenamentos de dados personalizados de vários formatos (incluindo arrays multidimensionais).

A Biblioteca Padrão MQL5 (em termos de arrays de dados) é colocada no diretório de trabalho do terminal, na pasta Include\Arrays.

Classe	Descrição
CArray	Classe base do array de dados dinâmico
CArrayChar	Array dinâmico de variáveis do tipo char ou uchar
CArrayShort	Array dinâmico de variáveis do tipo short ou ushort
CArrayInt	Array dinâmico de variáveis do tipo int ou uint
CArrayLong	Array dinâmico de variáveis do tipo long ou ulong
CArrayFloat	Array dinâmico de variáveis do tipo float
CArrayDouble	Array dinâmico de variáveis do tipo double
CArrayString	Array dinâmico de variáveis do tipo string
CArrayObj	Array dinâmico de ponteiros CObject
CList	Oferece a capacidade de trabalhar com uma lista de situações do CObject e seu descendente
CTreeNode	Oferece a capacidade de trabalhar com os nós da árvore binária Ctree
CTree	Oferece a capacidade de trabalhar com a árvore binária das instâncias de classe CTreeNode e seus descendentes

© 2000-2024, MetaQuotes Ltd.

CArray

CArray é uma classe base de array dinâmica de variáveis.

Descrição

Classe CArray possibilita trabalhar com um array dinâmico de variáveis na parte dos atributos de gerenciamento de alocação da memória, ordenação e trabalho com arquivos.

Declaração

```
class CArray : public CObject
```

Título

```
#include <Arrays\Array.mqh>
```

Hierarquia de herança

CObject

CArray

Descendentes diretos

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

Métodos de classe

Atributos	
Step	Obtém o tamanho do incremento de passo do array
Step	Defina o tamanho do incremento do array
Total	Obtém o número de elementos no array
Available	Obtém o número de elementos livres do array que estão disponíveis sem alocação de memória adicional
Max	Obtém o tamanho máximo possível do array sem realocação de memória
IsSorted	Obtém assinatura do array ordenado para uma opção específica
SortMode	Obtém a versão do array de classificação
Métodos claros	
Clear	Exclui todos os elementos do array, sem liberação de memória
Classificar métodos	
Sort	Classifica um array para uma opção específica
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Step

Define o tamanho do incremento do array.

```
int  Step() const
```

Valor do Retorno

Tamanho de incremento do array.

Exemplo

```
//--- example for CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get resize step
   int step=array.Step();
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Step

Define o tamanho do incremento do array.

```
bool  Step(
   int  step      // step
   )
```

Parâmetros

step

[in] Os novos valores dos incrementos dos passos no tamanho do array.

Valor do Retorno

Verdadeiro se obteve êxito, falso - se houve uma tentativa de estabelecer um passo menor ou igual a zero.

Exemplo

```
//--- example for CArray::Step(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- set resize step
   bool result=array.Step(1024);
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Total

Obtém o número de elementos no array.

```
int  Total() const;
```

Valor do Retorno

Número de elementos no array.

Exemplo

```
//--- example for CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check total
   int total=array.Total();
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Available

Obtém o número de elementos livres do array que estão disponíveis sem alocação de memória adicional.

```
int  Available() const
```

Valor do Retorno

Número de elementos livres do array que estão disponíveis sem alocação de memória adicional.

Exemplo

```
//--- example for CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check available
   int available=array.Available();
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Max

Obtém o tamanho máximo possível do array sem realocação de memória.

```
int  Max() const
```

Valor do Retorno

O tamanho máximo possível do array sem realocação de memória.

Exemplo

```
//--- example for CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check maximum size
   int max=array.Max();
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

IsSorted

Obtém assinatura do array ordenado para uma opção específica.

```
bool  IsSorted(
   int  mode=0       // Sorting mode
   ) const
```

Parâmetros

mode=0

[in] Versão de teste da classificação.

Valor do Retorno

Flag da lista ordenada. Se a lista é classificada pela versão específica - verdadeiro, do contrário - falso.

Observação

O sinal do array ordenado não pode ser alterado diretamente. O sinal classificado define o método Sort () e resetará qualquer métodos para adicionar / inserir exceto o InsertSort (...).

Exemplo

```
//--- example for CArray::IsSorted()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check sorted
   if(array.IsSorted())
     {
      //--- use methods for sorted array
      //--- ...
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SortMode

Obtém a versão do array de classificação.

```
int  SortMode() const;
```

Valor do Retorno

Modo de classificação.

Exemplo

```
//--- example for CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check sort mode
   int sort_mode=array.SortMode();
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Exclui todos os elementos do array, sem liberação de memória.

```
void  Clear()
```

Valor do Retorno

Nenhum.

Exemplo

```
//--- example for CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Object create error");
 return;
 }
 //--- use array
 //--- ...
 //--- clear array
 array.Clear();
 //--- delete array
 delete array;
 }
```

© 2000-2024, MetaQuotes Ltd.

Sort

Classifica um array para uma opção específica.

```
void  Sort(
   int  mode=0       // Sorting mode
   )
```

Parâmetros

mode=0

[in] Modo de classificação do array.

Valor do Retorno

Não tem.

Observação

Classificação de um array é sempre ascendente. Para arrays dos tipos de dados primitivos (CArrayChar, CArrayShort, etc.), o modo de parâmetro não é usado. Para CArrayObj, classificação multivariada deve ser implementada na classe derivada, método Sort (int).

Exemplo

```
//--- example for CArray::Sort(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- sorting by mode 0
   array.Sort(0);
   //--- use array
   //--- ...
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArray::Save(int)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArray::Load(...)
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayChar

CArrayChar é uma classe de array dinâmico de variáveis do tipo char ou uchar.

Descrição

A classe CArrayChar posibilita o trabalho com um array dinâmico de variáveis do tipo char ou uchar. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayChar : public CArray
```

Título

```
#include <Arrays\ArrayChar.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayChar

Métodos de classe

Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array
AssignArray	Copia os elementos array a partir de outro array
Métodos para modificar	

© 2000-2024, MetaQuotes Ltd.

Controle de memória	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Métodos para array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual a amostra em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayChar::Reserve(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayChar::Resize(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   char  element      // Element to add
   )
```

Parâmetros

element

[in] o valor do elemento para adicionar ao array.

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayChar::Add(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const char&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayChar::AddArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayChar*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro a uma instância de classe CArrayChar - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayChar::AddArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   char  element,      // Element to insert
   int   pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayChar::Insert(char,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos de outro array com uma posição específica.

```
bool  InsertArray(
   const char&  src[],      // Source array
   int           pos        // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayChar::InsertArray(const char &[],int)
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos de outro array com uma posição específica.

```
bool  InsertArray(
   CArrayChar*  src,      // Pointer to the source
   int           pos      // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayChar - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayChar::InsertArray(const CArrayChar*,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const char&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayChar::AssignArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayChar*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayChar - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayChar::AssignArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int   pos,          // Position
   char  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Novo elemento de valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayChar::Update(int,char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,'A'))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento array movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayChar::Shift(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Remove posição do elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayChar::Delete(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Positions of the last element
   )
```

Parâmetros

from

[in] Posição da primeira remoção do elemento no array.

to

[in] Posição do último elemento deletado no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayChar::DeleteRange(int,int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
char  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna CHAR_MAX se houver uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

CHAR_MAX pode ser um valor válido de um elemento array, então havendo um valor, verifique sempre o último código de erro.

Exemplo

```
//--- example for CArrayChar::At(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error of reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const char&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] A referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayChar::CompareArray(const char &[])
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const CArrayChar*  src      // Pointer to the sources
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayChar- elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayChar::CompareArray(const CArrayChar*)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   char  element      // Element to insert
   )
```

Parâmetros

element

[in] O valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayChar::InsertSort(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort('A'))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::Search(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search('A')!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchGreat(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchLess(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual ao modelo em array ordenado

```
int  SearchGreatOrEqual(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchGreatOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchLessOrEqual(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual('A')!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchFirst(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst('A')!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayChar::SearchLast(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast('A')!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Procura por elemento igual a amostra no array

```
int  SearchLinear(
   char  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayChar::SearchLinear(char)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear('A')!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - se concluído com êxito, falso - se com erro.

Exemplo

```
//--- example for CArrayChar::Save(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - se concluído com êxito, falso - se com erro.

Exemplo

```
//--- example for CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo da ID do array (por CArrayChar - 77).

Exemplo

```
//--- example for CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayShort

CArrayShorté uma classe de array dinâmico de variáveis do tipo short ou ushort.

Descrição

Classe CArrayShort posibilita o trabalho com um array dinâmico de variáveis do tipo short ou ushort. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayShort : public CArray
```

Título

```
#include <Arrays\ArrayShort.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayShort

Métodos de classe

Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	

© 2000-2024, MetaQuotes Ltd.

Controle de memória	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayShort::Resize(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   short  element      // Element to add
   )
```

Parâmetros

element

[in] O valor do elemento para adicionar ao array.

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayShort::Add(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const short&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayShort::AddArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayShort*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayShort - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayShort::AddArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   short  element,      // Element to insert
   int    pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayShort::Insert(short,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const short&  src[],      // Source array
   int            pos         // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayShort::InsertArray(const short &[],int)
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayShort*  src,      // Pointer to the source
   int            pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayShort - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayShort::InsertArray(const CArrayShort*,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const short&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayShort::AssignArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayShort*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayShort - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayShort::AssignArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int    pos,          // Position
   short  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Novo elemento de valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayShort::Update(int,short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,100))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Positions
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do elemento array movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayShort::Shift(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Remove posição do elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayShort::Delete(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of the lase element
   )
```

Parâmetros

from

[in] Posição da primeira remoção do elemento no array.

to

[in] Posição do último elemento deletado no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayShort::DeleteRange(int,int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
short  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna SHORT_MAX se houver uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

SHORT_MAX pode ser um valor válido de um elemento array, então havendo um valor, verifique sempre o último código de erro.

Exemplo

```
//--- example for CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error of reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const short&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] A referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayShort::CompareArray(const short &[])
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const CArrayShort*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayShort - elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayShort::CompareArray(const CArrayShort*)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   short  element      // Element to insert
   )
```

Parâmetros

element

[in] O valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayShort::InsertSort(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(100))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::Search(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search(100)!=-1) printf("Element found");
   else                      printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchGreat(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchLess(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchGreatOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual(100)!=-1) printf("Element found");
   else                                  printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchLessOrEqual(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual(100)!=-1) printf("Element found");
   else                                 printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchFirst(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst(100)!=-1) printf("Element found");
   else                           printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayShort::SearchLast(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast(100)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Procura por elemento igual a amostra no array

```
int  SearchLinear(
   short  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayShort::SearchLinear(short)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear(100)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayShort::Save(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID do array (por CArrayShort - 82).

Exemplo

```
//--- example for CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayInt

CArrayInt é uma classe de array dinâmico de variáveis do tipo int ou uint.

Descrição

Classe CArrayInt posibilita o trabalho com um array dinâmico de variáveis do tipo int ou uint. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayInt : public CArray
```

Título

```
#include <Arrays\ArrayInt.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayInt

Descendentes diretos

CSpreadBuffer

Métodos de classe

Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array

© 2000-2024, MetaQuotes Ltd.

Controle de memória	
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayInt::Reserve(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Number
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayInt::Resize(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   int  element      // Element to add
   )
```

Parâmetros

element

[in] o valor do elemento para adicionar ao array.

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayInt::Add(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const int&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayInt::AddArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayInt*  src      // Pointer to the source
   )
```

Parâmetros

src

Ponteiro para uma instância de classe CArrayInt - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayInt::AddArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   int  element,      // Element to insert
   int  pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayInt::Insert(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const int&  src[],      // Source array
   int          pos         // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayInt::InsertArray(const int &[],int)
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayInt*  src,      // Pointer to the source
   int          pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayInt - elementos fontes para inserir.

pos

[in] Posição no array para inserir.

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayInt::InsertArray(const CArrayInt*,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const int&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayInt::AssignArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayInt*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayInt - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayInt::AssignArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    //--- arrays is identical
    //--- delete source array
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
   }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int  pos,          // Position
   int  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar.

element

[in] Novo elemento de valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayInt::Update(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,10000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do elemento array movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayInt::Shift(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Remove posição do elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayInt::Delete(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of the last element
   )
```

Parâmetros

from

[in] Posição da primeira remoção do elemento no array.

to

[in] Posição do último elemento deletado no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayInt::DeleteRange(int,int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
int  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna INT_MAX se houver uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

INT_MAX pode ser um valor válido de elemento de array, assim havendo um valor, verificar sempre o último código de erro.

Exemplo

```
//--- example for CArrayInt::At(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error of reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const int&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] A referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayInt::CompareArray(const int &[])
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const CArrayInt*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayInt - elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayInt::CompareArray(const CArrayInt*)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   int  element      // Element to insert
   )
```

Parâmetros

element

[in] o valor do elemento a ser inserido dentro de um array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayInt::InsertSort(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(10000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::Search(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search(10000)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::SearchGreat(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::SearchLess(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   int  element      // Element to search
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::SearchGreatOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::SearchLessOrEqual(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt:: SearchFirst(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst(10000)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayInt::SearchLast(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast(10000)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Procura por elemento igual a amostra no array

```
int  SearchLinear(
   int  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayInt::SearchLinear(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear(10000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayInt::Save(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID do array(por CArrayInt - 82).

Exemplo

```
//--- example for CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayLong

CArrayLong é uma classe de array dinâmico de variáveis do tipo long ou ulong.

Descrição

Classe CArrayLong posibilita o trabalho com um array dinâmico de variáveis do tipo long ou ulong. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayLong : public CArray
```

Título

```
#include <Arrays\ArrayLong.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayLong

Descendentes diretos

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

Métodos de classe

Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array

© 2000-2024, MetaQuotes Ltd.

Controle de memória	
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayLong::Resize(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   long  element      // Element to add
   )
```

Parâmetros

element

[in] O valor do elemento para adicionar ao array.

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayLong::Add(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const long&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayLong::AddArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayLong*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayLong - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayLong::AddArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   long  element,      // Element to insert
   int   pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayLong::Insert(long,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const long&  src[],      // Source array
   int           pos         // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayLong::InsertArray(const long &[],int)
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayLong*  src,      // Pointer to the source
   int           pos      // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayLong - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayLong::InsertArray(const CArrayLong*,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const long&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayLong::AssignArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayLong*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayLong - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayLong::AssignArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int   pos,          // Position
   long  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Novo elemento de valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayLong::Update(int,long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,1000000))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do elemento array movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayLong::Shift(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Remove posição do elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayLong::Delete(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of the last element
   )
```

Parâmetros

from

[in] Posição da primeira remoção do elemento no array.

to

[in] Posição do último elemento deletado no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayLong::DeleteRange(int,int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
long  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna LONG_MAX se houver uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

LONG_MAX pode ser um valor válido de elemento de array, assim havendo um valor, verificar sempre o último código de erro.

Exemplo

```
//--- example for CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Error reading from the array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const long&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] A referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayLong::CompareArray(const long &[])
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArrayconst

Compara o array com outro array.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayLong - elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayLong::CompareArray(const CArrayLong*)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   long  element      // Element to insert
   )
```

Parâmetros

element

[in] O valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayLong::InsertSort(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(1000000))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual ao modelo apresentado no array ordenado.

```
int  Search(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::Search(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search(1000000)!=-1) printf("Element found");
   else                          printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de amostra maior, no array ordenado.

```
int  SearchGreat(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchGreat(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchLess(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual ao modelo no array ordenado.

```
int  SearchGreatOrEqual(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchGreatOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");
   else                                      printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual ao modelo no array ordenado.

```
int  SearchLessOrEqual(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchLessOrEqual(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo no array ordenado.

```
int  SearchFirst(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchFirst(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst(1000000)!=-1) printf("Element found");
   else                               printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayLong::SearchLast(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast(1000000)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Pesquisas para o elemento igual ao modelo no array.

```
int  SearchLinear(
   long  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayLong::SearchLinear(long)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear(1000000)!=-1) printf("Element found");
   else                                printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - se concluído com êxito, falso - se com erro.

Exemplo

```
//--- example for CArrayLong::Save(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - se concluído com êxito, falso - se com erro.

Exemplo

```
//--- example for CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID da matriz (por CArrayLong - 84).

Exemplo

```
//--- example for CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayFloat

CArrayFloat é uma classe de array dinâmico de variáveis do tipo float.

Descrição

A classe CArrayFloat posibilita o trabalho com um array dinâmico de variáveis do tipo float. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayFloat : public CArray
```

Título

```
#include <Arrays\ArrayFloat.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayFloat

Métodos de classe

Atributos	
Delta	Define a tolerância de comparação
Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array

© 2000-2024, MetaQuotes Ltd.

Atributos	
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Delta

Define a comparação de tolerância.

```
void  Delta(
   float  delta      // Tolerance
   )
```

Parâmetros

delta

[in] o novo valor de admissão de comparação.

Valor do Retorno

Nenhum

Observação

Admissão de comparação utilizado na pesquisa. Os valores são considerados iguais se a diferença for menor ou igual a tolerância. A tolerância padrão é 0.0.

Exemplo

```
//--- example for CArrayFloat::Delta(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- set compare variation
   array.Delta(0.001);
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayFloat::Reserve(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayFloat::Resize(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   float  element      // Element to add
   )
```

Parâmetros

element

[in] O valor do elemento para adicionar ao array.

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayFloat::Add(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const float&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayFloat::AddArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayFloat*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância da classe elementos CArrayFloat - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayFloat::AddArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   float  element,      // Element to insert
   int    pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayFloat::Insert(float,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos de outro array com uma posição específica.

```
bool  InsertArray(
   const float&  src[],      // Source array
   int            pos         // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayFloat::InsertArray(const float &[],int)
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayFloat*  src,      // Pointer to the source
   int            pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayFloat - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayFloat::InsertArray(const CArrayFloat*,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const float&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayFloat::AssignArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayFloat*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayFloat - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayFloat::AssignArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int    pos,          // Position
   float  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Novo elemento de valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayFloat::Update(int,float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do elemento array movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayFloat::Shift(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Remove posição do elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayFloat::Delete(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of last element
   )
```

Parâmetros

from

[in] Posição da primeira remoção do elemento no array.

to

[in] Posição do último elemento deletado no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayFloat::DeleteRange(int,int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
float  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna FLT_MAX se houve uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

FLT_MAX pode ser um valor válido de elemento de array, assim havendo um valor, verificar sempre o último código de erro.

Exemplo

```
//--- example for CArrayFloat::At(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const float&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] A referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayFloat::CompareArray(const float &[])
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArrayconst

Copia os elementos array a partir de outro array.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayFloat - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayFloat::CompareArray(const CArrayFloat*)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   float  element      // Element to insert
   )
```

Parâmetros

element

[in] O valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayFloat::InsertSort(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::Search(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::SearchGreat(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat:: SearchLess(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::SearchGreatOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::SearchLessOrEqual(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::SearchFirst(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayFloat::SearchLast(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Procura por elemento igual a amostra no array

```
int  SearchLinear(
   float  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayFloat::SearchLinear(float)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayFloat::Save(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo da ID do array (para CArrayFloat - 87).

Exemplo

```
//--- example for CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayDouble

CArrayDouble é uma classe de array dinâmico de variáveis do tipo double.

Descrição

Classe CArrayDouble posibilita o trabalho com um array dinâmico de variáveis do tipo double. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayDouble : public CArray
```

Título

```
#include <Arrays\ArrayDouble.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble

Descendentes diretos

CDoubleBuffer

Métodos de classe

Atributos	
Delta	Define a tolerância de comparação
Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma liberação de memória completa
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica

© 2000-2024, MetaQuotes Ltd.

Atributos	
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Procurar por min/max	
Minimum	Obtém o índice do menor elemento do array no intervalo determinado
Maximum	Obtém o índice maior elemento do matriz no intervalo determinado
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
SearchLinear	Procura o elemento igual a amostra no array

© 2000-2024, MetaQuotes Ltd.

Atributos	
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Delta

Define a comparação de tolerância.

```
void  Delta(
   double  delta      // Tolerance
   )
```

Parâmetros

delta

[in] O novo valor da admissão de comparação.

Valor do Retorno

Nenhum

Observação

Admissão de comparação utilizado na pesquisa. Os valores são considerados iguais se a diferença for menor ou igual a tolerância. A tolerância padrão é 0.0.

Exemplo

```
//--- example for CArrayDouble::Delta(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- set compare variation
   array.Delta(0.001);
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayDouble::Reserve(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Novo tamanho do array.

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayDouble::Resize(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   double  element      // Element to add
   )
```

Parâmetros

element

[in] o valor do elemento para adicionar ao array

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayDouble::Add(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const double&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayDouble::AddArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayDouble*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância da classe CArrayDouble - elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayDouble::AddArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   double  element,      // Element to insert
   int     pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayDouble::Insert(double,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const double&  src[],      // Source array
   int            pos         // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayDouble::InsertArray(const double &[],int)
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayDouble*  src,      // Pointer to the source
   int            pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância da classe CArrayDouble - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayDouble::InsertArray(const CArrayDouble*,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const double&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayDouble::AssignArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayDouble*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância da classe CArrayDouble - elemento fonte para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayDouble::AssignArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int     pos,          // Position
   double  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Novo Valor.

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayDouble::Update(int,double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,100.0))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do array de elemento movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayDouble::Shift(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] posição remove o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayDouble::Delete(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of last element
   )
```

Parâmetros

from

[in] Posição do primeiro remove o elemento no array.

to

[in] Posição do último exclue o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayDouble::DeleteRange(int,int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
double  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna DBL_MAX se houve uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

DBL_MAX pode ser um valor válido de elemento de array, assim havendo um valor, verificar sempre o último código de erro.

Exemplo

```
//--- example for CArrayDouble::At(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  Error reading from the array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const double&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayDouble::CompareArray(const double &[])
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const CArrayDouble*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância da classe CArrayDouble - elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayDouble::CompareArray(const CArrayDouble*)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Minimum

Obtém o índice do elemento menor de array no intervalo específico.

```
int  Minimum(
   int  start,      // starting index
   int  count       // number of elements to proceed
   ) const
```

Parâmetros

start

[in] Iniciando índice do array.

count

[in] Número de elementos para proceder.

Valor de retorno

Índice do elemento menor de array no intervalo específico.

© 2000-2024, MetaQuotes Ltd.

Maximum

Obtém o índice do elemento maior de array no intervalo específico.

```
int  Maximum(
   int  start,      // starting index
   int  count       // number of elements to proceed
   ) const
```

Parâmetros

start

[in] Iniciando índice do array.

count

[in] Número de elementos para proceder.

Valor de retorno

Índice do elemento maior de array no intervalo específico.

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   double  element      // Element to insert
   )
```

Parâmetros

element

[in] o valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayDouble::InsertSort(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(100.0))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::Search(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search(100.0)!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::SearchGreat(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble:: SearchLess(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::SearchGreatOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::SearchLessOrEqual(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::SearchFirst(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst(100.0)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayDouble::SearchLast(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast(100.0)!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Pesquisa o elemento igual da amostra no array ordenado.

```
int  SearchLinear(
   double  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayDouble::SearchLinear(double)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear(100.0)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayDouble::Save(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID do array (por CArrayDouble - 87).

Exemplo

```
//--- example for CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayString

CArrayString é uma classe de array dinâmico de variáveis do tipo string.

Descrição

Classe CArrayString posibilita o trabalho com um array dinâmico de variáveis do tipo string. A classe possibilita adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Declaração

```
class CArrayString : public CArray
```

Título

```
#include <Arrays\ArrayString.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayString

Métodos de classe

Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Define um novo tamanho (mais pequeno) do array
Métodos para adicionar	
Add	Adiciona um elemento para o final do array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
ddArray	Adiciona ao final dos elementos array a partir de um outro array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	

© 2000-2024, MetaQuotes Ltd.

Controle de memória	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
CompareArray	Compara o array com outro array
Operação do array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao da amostra no array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual a amostra no array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual a amostra no array ordenado
SearchFirst	Localiza o primeiro elemento igual a amostra no array ordenado
SearchLast	Localiza o último elemento igual a amostra no array ordenado
SearchLinear	Procura o elemento igual a amostra no array
Entrada/saída	
virtual Save	Salva array de dados no arquivo
virtual Load	Carrega array de dados a partir de um arquivo
virtual Type	Obtém o identificador do tipo do array

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayString::Reserve(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reserve memory
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Nova tamanho do array

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterando o tamanho do array permite uma utilização ótima do uso de memória. Elementos supérfluos perdidos à direita. Para reduzir a fragmentação da memória, o tamanho do array é alterado com um passo dado previamente através do método de Step (int), ou 16 (padrão).

Exemplo

```
//--- example for CArrayString::Resize(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma liberação de memória completa.

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Exemplo

```
//--- example for CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   string  element      // Element to add
   )
```

Parâmetros

element

[in] o valor do elemento para adicionar ao array

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Exemplo

```
//--- example for CArrayString::Add(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const string&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayString::AddArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayString*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayString - a fonte de elementos a acrescentar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Exemplo

```
//--- example for CArrayString::AddArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- add another array
   if(!array.AddArray(src))
     {
      printf("Array addition error");
      delete src;
      delete array;
      return;
     }
   //--- delete source array
   delete src;
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   string  element,      // Element to insert
   int     pos           // Position
   )
```

Parâmetros

element

[in] O valor do elemento para ser inserido no array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayString::Insert(string,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const string&  src[],      // Source array
   int             pos        // Position
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para inserir

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayString::InsertArray(const string &[],int)
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   CArrayString*  src,      // Pointer to the source
   int             pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayString - a fonte de elementos para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Exemplo

```
//--- example for CArrayString::InsertArray(const CArrayString*,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- insert another array
   if(!array.InsertArray(src,0))
     {
      printf("Array inserting error");
      delete src;
      delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
    }
  //--- delete source array
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const string&  src[]      // Source array
   )
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayString::AssignArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayString*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayString - fonte de elementos para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Exemplo

```
//--- example for CArrayString::AssignArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete src;
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int     pos,          // Position
   string  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Elemento de novo valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Exemplo

```
//--- example for CArrayString::Update(int, string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,"ABC"))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do array de elemento movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayString::Shift(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] posição remove o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Exemplo

```
//--- example for CArrayString::Delete(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete element
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of last element
   )
```

Parâmetros

from

[in] Posição do primeiro remove o elemento no array.

to

[in] Posição do último exclue o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Exemplo

```
//--- example for CArrayString::DeleteRange(int,int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
string  At(
   int  pos      // Position
   ) const
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento em caso de sucesso. Retorna aspas (") se houver uma tentativa de obter um elemento de posições não existentes (o último erro ERR_OUT_OF_RANGE).

Observação

Duas aspas ("") pode ser um valor válido de um elemento do array, então havendo um valor, verifique sempre o último código de erro

Exemplo

```
//--- example for CArrayString::At(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result==""  && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Error reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
```

© 2000-2024, MetaQuotes Ltd.

```
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const string&  src[]      // Source array
   ) const
```

Parâmetros

src[]

[in] Referência a um array de elementos fontes para comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayString::CompareArray(const string &[])
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArrays(
   const CArrayString*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayString - a fonte de elementos de comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- add source arrays elements
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   string  element      // Element to insert
   )
```

Parâmetros

element

[in] o valor do elemento a ser inserido em array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Exemplo

```
//--- example for CArrayString::InsertSort(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort("ABC"))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString::Search(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.Search("ABC")!=-1) printf("Element found");
   else                        printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString::SearchGreat(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreat("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString:: SearchLess(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLess("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString:: SearchGreatOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString:: SearchLessOrEqual(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Element found");
   else                                   printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString:: SearchFirst(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchFirst("ABC")!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayString:: SearchLast(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- search element
   if(array.SearchLast("ABC")!=-1) printf("Element found");
   else                            printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLinear

Procura por elemento igual a amostra no array

```
int  SearchLinear(
   string  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

A posição do elemento encontrado, se com sucesso. Igual a -1 se o elemento não foi encontrado.

Observação

O método usa o algoritmo de busca linear (ou busca sequencial) para arrays não classificados.

Exemplo

```
//--- example for CArrayString::SearchLinear(string)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- search element
   if(array.SearchLinear("ABC")!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayString::Save(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2024, MetaQuotes Ltd.

```
    }
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega array de dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use arrays elements
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

© 2000-2024, MetaQuotes Ltd.

```
      }
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID do array (por CArrayString - 89).

Exemplo

```
//--- example for CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CArrayObj

CArrayObj é uma classe de array dinâmico de ponteiros para instâncias de CObject e seus herdeiros.

Descrição

A classe CArrayObj possibilita trabalhar com um array dinâmico de ponteiros para instâncias de CObject e seus herdeiros. Isto dá a possibilidade de trabalhar como arrays dinâmicos multidimensionais de tipos de dados primitivos e para estruturas de dados organizados mais difíceis.

Na classe implementada existe a possibilidade de adicionar/inserir/excluir elementos em: array, ordenação de array e pesquisando em array ordenado. Além disto, implementa métodos para trabalhar com o arquivo.

Existem certas sutilezas na classe CArrayObj.

Declaração

```
class CArrayObj : public CArray
```

Título

```
#include <Arrays\ArrayObj.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj

Descendentes diretos

CIndicators, CSeries

Método de Classe

Atributos	
FreeMode	Obtém o flag de gerenciamento de memória
FreeMode	Define o flag de gerenciamento de memória
Controle de memória	
Reserve	Memória alocada para aumentar o tamanho do array
Resize	Define um novo tamanho (mais pequeno) do array
Shutdown	Limpa o array com uma isenção total do array de memória (não do elemento).
Métodos para adicionar	
Add	Adiciona um elemento para o final do array

© 2000-2024, MetaQuotes Ltd.

Atributos	
ddArray	Adiciona um elemento para o final do array
Insert	Insere um elemento no array para uma posição específica
InsertArray	Insere um array de elementos a partir de outro array com posição específica
AssignArray	Copia os elementos array a partir de outro array
Métodos de atualização	
Update	Altera o elemento na posição do array determinado
Shift	Desloca um item a partir de uma dada posição no array para um deslocamento determinado
Métodos para excluir	
Detach	Obtém o elemento a partir da posição determinada e remove-o a partir do array
Delete	Remove o elemento a partir de uma posição específica no array
DeleteRange	Exclui um grupo de elementos a partir de uma posição específica no array
Clear	Remove todos os elementos do array, sem a liberação do array de memória
Métodos de acesso	
At	Obtém o elemento a partir de uma posição dada no array
Métodos para comparar	
CompareArray	Compara o array com outro array
Operações de array ordenado	
InsertSort	Elemento de inserções em array ordenado
Pesquisa	Procura por um elemento igual ao modelo em array ordenado
SearchGreat	Procura por um elemento de mais amostras em array ordenado
SearchLess	Procura por um elemento menor do que a amostra em array ordenado
SearchGreatOrEqual	Procura por um elemento maior ou igual ao modelo em array ordenado
SearchLessOrEqual	Procura por um elemento menor ou igual ao modelo apresentado em array ordenado
SearchFirst	Localiza o primeiro elemento igual ao modelo em array ordenado
SearchLast	Localiza o último elemento igual ao modelo em array ordenado
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
Save	Salva array de dados no arquivo
Load	Carrega array de dados a partir de um arquivo
Type	Obtém o identificador do tipo do array

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

A aplicação prática dos arrays são descendentes da classe CObject (incluindo todas as classes da biblioteca padrão).

Por exemplo, considere as opções para o array bidimensional:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- create array
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
//--- create subarrays
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Object create error");
         return;
        }
      //--- fill array
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
```

© 2000-2024, MetaQuotes Ltd.

```
        }
      array.Add(sub_array);
     }
//--- create array OK
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- use array element
        }
     }
   delete array;
  }
```

Sutilezas

A classe tem um mecanismo para controlar a dinâmica de memória, por isso tenha cuidado ao trabalhar com elementos do array.

Mecanismo de gerenciamento de memória pode ser ligado / desligado através do método FreeMode (bool). Por padrão, o mecanismo é ativado.

Por conseguinte, existem duas opções para lidar com a classe CArrayObj classe:

1. Mecanismo de gerenciamento de memória está ativada. (padrão)

Neste caso, a CArrayObj assume a responsabilidade de liberar os elementos de memória após a sua remoção do array. Neste programa o usuário não deve liberar os elementos do array.

Exemplo

```
   int i;
//--- Create an array
   CArrayObj *array=new CArrayObj;
//--- Fill array elements
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action with an element
      . . .
     }
//--- Remove the array with the elements
   delete array;
```

2. Mecanismo de gerenciamento de memória está desativado.

© 2000-2024, MetaQuotes Ltd.

Neste caso, a CArrayObj não assume a responsabilidade para liberar os elementos de memória após a sua remoção do array. Neste programa o usuário deve liberar os elementos do array.

Exemplo

```
   int i;
//--- Create an array
   CArrayObj *array=new CArrayObj;
//--- Disable the mechanism of memory management
   array.FreeMode(false);
//--- Fill array elements
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- Action with an element
      . . .
     }
//--- Remove array elements
   while(array.Total()) delete array.Detach();
//--- Remove empty array
   delete array;
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Obtém o flag de gerenciamento de memória.

```
bool  FreeMode() const
```

Valor do Retorno

Flag de gerenciamento de memória .

Exemplo

```
//--- example for CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get free mode flag
   bool array_free_mode=array.FreeMode();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Define o flag de gerenciamento de memória.

```
void  FreeMode(
   bool  mode       // New flag
   )
```

Parâmetros

mode

[in] O novo valor do flag de gerenciamento de memória.

Valor do Retorno

Nenhum.

Observação

Definir o flag de gerenciamento de memória - um ponto importante no uso da classe CArrayObj. Desde que os elementos do array são ponteiros para objetos dinâmicos, é importante determinar o que fazer com eles quando remover o array.

Se o flag estiver definido, remove um elemento a partir do array, o elemento é automaticamente excluído pelo operador "delete". Se o flag não está definido, é assumido que um ponteiro para o objeto excluído ainda está em algum lugar no programa do usuário e será aliviado dele (do programa) então.

Se o usuário resetar o flag de gerenciamento de memória, deve compreender a sua responsabilidade para a remoção do array antes da conclusão do programa, porque de outra forma, a memória liberada não é ocupada pelos elementos quando eles criam um novo operador.

Se o usuário não resetar o flag de gerenciamento de memória, há um outro "recife", ou seja, grandes quantidades de dados eventualmente poderiam até mesmo quebrar o seu terminal.

Usando ponteiros e array armazenados em algum lugar das variáveis locais, após a remoção do array irá levar a um erro crítico e trava o programa do usuário. Por padrão, o flag de gerenciamento de memória está definido, ou seja, a classe do array é responsável por liberar os elementos de memória.

Exemplo

```
//--- example for CArrayObj::FreeMode(bool)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- reset free mode flag
   array.FreeMode(false);
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Reserve

Memória alocada para aumentar o tamanho do array.

```
bool  Reserve(
   int  size      // Number
   )
```

Parâmetros

size

[in] O número de elementos adicionais do array.

Valor do Retorno

verdadeiro se com sucesso; falso - se houve uma tentativa de buscar com quantidade menor ou igual a zero, ou se o array não aumentou.

Observação

Para reduzir a fragmentação da memória, o aumento do tamanho do array é feito com um passo dado anteriormente através do método de Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Reserve error");
      delete array;
      return;
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Resize

Define um novo tamanho (menor) do array.

```
bool  Resize(
   int  size      // Size
   )
```

Parâmetros

size

[in] Nova tamanho do array

Valor do Retorno

Verdadeiro se com sucesso; falso - se houve uma tentativa de definir o tamanho inferior a zero.

Observação

Alterar o tamanho do array permite o uso de memória na maneira ideal. Excesso de elementos localizados à direita são perdidos. A memória para elementos perdidos é liberada ou não, dependendo do modo de gerenciamento de memória.

Para reduzir a fragmentação da memória, alterar o tamanho do array com um passo dado previamente pelo método Step (int), ou 16 (por padrão).

Exemplo

```
//--- example for CArrayObj::Resize(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- resize array
   if(!array.Resize(10))
     {
      printf("Resize error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos do array sem a liberação do array de memória.

```
void  Clear()
```

Valor do Retorno

Não tem.

Observação

Se ativado o gerenciamento de memória, a memória e os itens excluídos estarão isentos.

Exemplo

```
//--- example for CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- clear array
   array.Clear();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shutdown

Limpa o array com uma isenção total do array de memória (não do elemento).

```
bool  Shutdown()
```

Valor do Retorno

Verdadeiro se obteve êxito, falso - se ocorreu um erro.

Observação

Se ativado o gerenciamento de memória, a memória e os itens excluídos estarão isentos.

Exemplo

```
//--- example for CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shutdown array
   if(!array.Shutdown())
     {
      printf("Shutdown error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateElement

Cria um novo elemento do array na posição determinada.

```
bool  CreateElement(
   int  index      // Position
   )
```

Parâmetros

index

[in] posição em que deseja-se criar um novo elemento.

Valor do Retorno

verdadeiro se obteve êxito, falso - não pode criar o elemento.

Observação

Método CreateElement (int) na classe CArrayObj sempre retorna falso e não executa qualquer ação. Se necessário, em uma classe derivada, o método CreateElement (int) deve ser implementado.

Exemplo

```
//--- example for CArrayObj::CreateElement(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- fill array
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Element create error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final do array.

```
bool  Add(
   CObject*  element      // Element to add
   )
```

Parâmetros

element

[in] o valor do elemento para adicionar ao array

Valor do Retorno

verdadeiro com sucesso, falso - não pode adicionar um elemento.

Observação

O elemento não é adicionado ao array, se o valor for inválido para ponteiro de transmissão (tais como NULL).

Exemplo

```
//--- example for CArrayObj::Add(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 arrays elements
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Element addition error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

ddArray

Adiciona ao final dos elementos array a partir de um outro array.

```
bool  AddArray(
   const CArrayObj *  src      // Pointer to the array-source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayDouble - fontes de elementos para adicionar.

Valor do Retorno

verdadeiro com sucesso, falso - se você não pode adicionar itens.

Observação

A adição de elementos do array no array é realmente copiar os ponteiros. Portanto, quando na chamada do método existe uma "barreira", então pode haver um ponteiro para um objeto dinâmico em mais de uma variável.

```
//--- example
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- Create a new instance CArrayObj
//--- Default memory management is turned on
CArrayObj *array=new CArrayObj;
//--- Add (copy) the elements of the source array
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- Perform erroneous actions
   switch(error)
     {
      case 0:
         //--- Remove the source array, without checking its memory management flag
         delete src;
         //--- Result:
         //--- It is possible to address an element by invalid pointer in the receive
         break;
      case 1:
         //--- Disable the mechanism of memory management in the source array
         if(src.FreeMode()) src.FreeMode(false);
         //--- But do not remove the source array
         //--- Result:
         //--- After removing the receiver array it is possible to address an element
         break;
```

© 2000-2024, MetaQuotes Ltd.

```
      case 2:
         //--- Disable the mechanism of memory management in the source array
         src.FreeMode(false);
         //--- Disable the mechanism of memory management in the receiver array
         array.FreeMode(false);
         //--- Result:
         //--- After the program termination get a "memory leak"
         break;
      }
   }
else
   {
    //--- Disable the mechanism of memory management in the source array
    if(src.FreeMode()) src.FreeMode(false);
    //--- Delete the source array
    delete src;
    //--- Result:
    //--- Addressing the receiver array element will be correct
    //--- Deleting the receiver array will lead to deleting its elements
   }
```

Exemplo

```
//--- example for CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
   {
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
      {
       printf("Object create error");
       return;
      }
    //--- create source array
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
      {
       printf("Object create error");
       delete array;
       return;
      }
    //--- reset free mode flag
    src.FreeMode(false);
    //--- fill source array
    //--- . . .
    //--- add another array
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!array.AddArray(src))
    {
     printf("Array addition error");
     delete src;
     delete array;
     return;
    }
  //--- delete source array without elements
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento no array para uma posição específica.

```
bool  Insert(
   CObject*  element,      // Element to insert
   int       pos           // Position
   )
```

Parâmetros

element

[in] o valor do elemento para ser inserido em um array

pos

[in] Posição no array para inserir

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Observação

O elemento não é adicionado ao array, se o valor for inválido para ponteiro de transmissão (tais como NULL).

Exemplo

```
//--- example for CArrayObj::Insert(CObject*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert elements
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Insert error");
         delete array;
         return;
        }
     }
   //--- use array
   //--- . . .
```

© 2000-2024, MetaQuotes Ltd.

```
    //--- delete array
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertArray

Insere um array de elementos a partir de outro array com posição específica.

```
bool  InsertArray(
   const CArrayObj*  src,      // Pointer to the source
   int               pos       // Position
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayObj - elementos fontes para inserir.

pos

[in] Posição no array para inserir

Valor do Retorno

Verdadeiro se com sucesso, falso - não pode colar os itens.

Observação

Veja: CArrayObj::AddArray(const CArrayObj*).

Exemplo

```
//--- example for CArrayObj::InsertArray(const CArrayObj*,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- reset free mode flag
   src.FreeMode(false);
   //--- fill source array
   //--- . . .
   //--- insert another array
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!array.InsertArray(src,0))
    {
     printf("Array inserting error");
     delete src;
     delete array;
     return;
    }
  //--- delete source array without elements
  delete src;
  //--- use array
  //--- . . .
  //--- delete array
  delete array;
}
```

© 2000-2024, MetaQuotes Ltd.

AssignArray

Copia os elementos array a partir de outro array.

```
bool  AssignArray(
   const CArrayObj*  src      // Pointer to the source
   )
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayObj - fonte de elementos para copiar.

Valor do Retorno

verdadeiro se com sucesso, falso - não é possível copiar os itens.

Observação

Se o receptor array AssignArray não está vazio, todos os seus elementos serão removidos do array e o flag de gerenciamento de memória, memória e os itens apagados serão liberados. O receptor Array é uma cópia exata da fonte do array. Veja também CArrayObj::AddArray(const CArrayObj*).

Exemplo

```
//--- example for CArrayObj::AssignArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- reset free mode flag
   src.FreeMode(false);
   //--- fill source array
   //--- . . .
   //--- assign another array
   if(!array.AssignArray(src))
     {
```

© 2000-2024, MetaQuotes Ltd.

```
      printf("Array assigned error");
      delete src;
      delete array;
      return;
     }
   //--- arrays is identical
   //--- delete source array without elements
   delete src;
   //--- use array
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Update

Altera o elemento no array de posição específica.

```
bool  Update(
   int       pos,          // Position
   CObject*  element       // Value
   )
```

Parâmetros

pos

[in] Posição do elemento no array para alterar

element

[in] Elemento de novo valor

Valor do Retorno

verdadeiro se com sucesso; falso - não pode alterar o elemento.

Observação

O elemento não muda se um parâmetro passar por um ponteiro inválido (isto é NULO). Se ativado o gerenciamento de memória, o espaço reservado de memória é liberado.

Exemplo

```
//--- example for CArrayObj::Update(int,CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- update element
   if(!array.Update(0,new CObject))
     {
      printf("Update error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Shift

Desloca um item a partir de uma dada posição no array para um deslocamento determinado.

```
bool  Shift(
   int  pos,        // Position
   int  shift       // Shift
   )
```

Parâmetros

pos

[in] Posição do array de elemento movido

shift

[in] O valor de deslocamento (tanto positivo como negativo).

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CArrayObj::Shift(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- shift element
   if(!array.Shift(10,-5))
     {
      printf("Shift error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Detach

Remove um item a partir de uma determinada posição no array.

```
CObject*  Detach(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Posição do item apreendido no array.

Valor do Retorno

Ponteiro para a remoção de elementos em caso de sucesso. Retorna NULL se não pode remover o elemento.

Observação

Quando removido a partir do elemento de array, não será removido em qualquer estado do flag de gerenciamento de memória. Ponteiro para o elemento de array retirado a partir dos elementos da liberação após uso.

Exemplo

```
//--- example for CArrayObj::Detach(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Detach error");
      delete array;
      return;
     }
   //--- use element
   //--- . . .
   //--- delete element
   delete object;
   //--- delete array
```

© 2000-2024, MetaQuotes Ltd.

```
    delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir de uma posição dada no array.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] posição remove o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Observação

Se ativado o gerenciamento de memória, a memória e os itens excluídos estarão isentos.

Exemplo

```
//--- example for CArrayObj::Delete(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteRange

Exclui um grupo de elementos a partir de uma posição dada no array.

```
bool  DeleteRange(
   int  from,      // Position of the first element
   int  to         // Position of last element
   )
```

Parâmetros

from

[in] Posição do primeiro remove o elemento no array.

to

[in] Posição do último exclue o elemento no array.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover elementos.

Observação

Se ativado o gerenciamento de memória, a memória e os itens excluídos estarão isentos.

Exemplo

```
//--- example for CArrayObj::DeleteRange(int,int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- delete elements
   if(!array.DeleteRange(0,10))
     {
      printf("Delete error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento a partir de uma posição dada no array.

```
CObject*  At(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Posição do elemento desejado no array.

Valor do Retorno

O valor do elemento, se bem sucedido. Retorna NULL se houver uma tentativa de obter um elemento de posição inexistente.

Exemplo

```
//--- example for CArrayObj::At(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add elements
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Error reading from array
         printf("Get element error");
         delete array;
         return;
        }
      //--- use element
      //--- . . .
     }
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareArray

Compara o array com outro array.

```
bool  CompareArray(
   const CArrayObj*  src      // Pointer to the source
   ) const
```

Parâmetros

src

[in] Ponteiro para uma instância de classe CArrayObj - a fonte de elementos de comparação.

Valor do Retorno

Verdadeiro se arrays são iguais, falso - se não.

Exemplo

```
//--- example for CArrayObj::CompareArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source array
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Object create error");
      delete array;
      return;
     }
   //--- fill source array
   //--- . . .
   //--- compare with another array
   int result=array.CompareArray(src);
   //--- delete arrays
   delete src;
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

InsertSort

Elemento de inserções no array ordenado

```
bool  InsertSort(
   CObject*  element      // Element to insert
   )
```

Parâmetros

element

[in] O valor do elemento a ser inserido num array ordenado

Valor do Retorno

verdadeiro se com sucesso, falso - não pode inserir o elemento.

Observação

O elemento não é adicionado ao array, se o valor for inválido para ponteiro de transmissão (tais como NULL).

Exemplo

```
//--- example for CArrayObj::InsertSort(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- insert element
   if(!array.InsertSort(new CObject))
     {
      printf("Insert error");
      delete array;
      return;
     }
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual a amostra em array ordenado.

```
int  Search(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj::Search(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.Search(sample)!=-1) printf("Element found");
   else                         printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreat

Procura por um elemento de mais amostras em array ordenado.

```
int  SearchGreat(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj::SearchGreat(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchGreat(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLess

Procura por um elemento menor do que a amostra no array ordenado.

```
int  SearchLess(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj:: SearchLess(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchLess(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchGreatOrEqual

Procura por um elemento maior ou igual da amostra no array ordenado.

```
int  SearchGreatOrEqual(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj::SearchGreatOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");
   else                                     printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLessOrEqual

Procura por um elemento menor ou igual a amostra no array ordenado.

```
int  SearchLessOrEqual(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj:: SearchLessOrEqual(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");
   else                                    printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchFirst

Localiza o primeiro elemento igual ao modelo em array ordenado.

```
int  SearchFirst(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj::SearchFirst(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchFirst(sample)!=-1) printf("Element found");
   else                              printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

SearchLast

Localiza o último elemento igual ao modelo no array ordenado.

```
int  SearchLast(
   CObject*  element      // Sample
   ) const
```

Parâmetros

element

[in] O elemento de amostra para pesquisar no array.

Valor do Retorno

Se a posição do elemento foi encontrado com sucesso. Igual a -1, se o item não foi encontrado.

Exemplo

```
//--- example for CArrayObj:: SearchLast(CObject*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- sort array
   array.Sort();
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete array;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(array.SearchLast(sample)!=-1) printf("Element found");
   else                             printf("Element not found");
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva array de dados no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abertura prévia pelo arquivo de função binária FileOpen (...).

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CArrayObj::Save(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add arrays elements
   //--- . . .
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Array carrega dados a partir de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para abrir mais cedo, com a função FileOpen (...), arquivo binário.

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Observação

Quando lê a partir do array de arquivo é criado cada elemento do método chamando CArrayObj :: CreateElement (int).

Exemplo

```
//--- example for CArrayObj::Load(int)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int        file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use arrays elements
   //--- . . .
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do array.

```
virtual int  Type() const
```

Valor do Retorno

Tipo de ID da matriz (por CArrayObj - 7778).

Exemplo

```
//--- example for CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get array type
   int type=array.Type();
   //--- delete array
   delete array;
  }
```

© 2000-2024, MetaQuotes Ltd.

CList

CList classe é uma classe de lista dinâmica de instâncias da classe CObject e seus herdeiros.

Descrição

Classe CList possibilita trabalhar com uma lista de instâncias de CObject e seus herdeiros. Na classe implementada existe a possibilidade de adicionar/inserir/excluir itens na lista, ordenar a lista e procurar na lista ordenada. Além disto, implementa métodos para trabalhar com o arquivo.

Existem certas sutilezas na classe CList. A classe tem um mecanismo para controlar a dinâmica de memória, por isso deve-se ter cuidado quando se trabalha com elementos da lista.

As sutilezas do mecanismo de gestão de memória são semelhantes às descritas no CArrayObj.

Declaração

```
class CList : public CObject
```

Título

```
#include <Arrays\List.mqh>
```

Hierarquia de herança

CObject
  CList

Métodos de classe

Atributos	
FreeMode	Obtém o flag de gerenciamento de memória ao excluir itens da lista.
FreeMode	Define o flag de gerenciamento de memória ao excluir itens na lista
Total	Obtém o número de elementos na lista
IsSorted	Obtém flag da lista ordenada
SortMode	Obtém a versão da ordenação
Criar métodos	
CreateElement	Cria um novo item à lista
Métodos para adicionar	
Add	Adiciona elemento para o final da lista
Insert	Elemento de inserções na lista na posição determinada
Métodos para excluir	
DetachCurrent	Remove um item a partir da posição atual da lista sem excluir o item "fisicamente"

© 2000-2024, MetaQuotes Ltd.

Atributos	
DeleteCurrent	Remove o elemento a partir da posição atual na lista
Delete	Remove o elemento a partir de posição determinada na lista
Clear	Remove todos os itens da lista
Navegação	
IndexOf	Obtém o índice do item da lista
GetNodeAtIndex	Obtém um item com o índice determinado da lista
GetFirstNode	Obtém o primeiro elemento da lista
GetPrevNode	Obtém o elemento anterior da lista
GetCurrentNode	Obtém o item atual da lista
GetNextNode	Obtém o próximo item na lista
GetLastNode	Obtém o último item
Métodos de ordenação	
Sort	Ordenar lista
MoveToIndex	Move o item da lista atual para uma posição determinada
Exchange	Altera elementos dos lugares da lista
Métodos para comparar	
CompareList	Compara a lista com outra lista
Os métodos de pesquisa	
Pesquisa	Procura por um elemento igual ao modelo de lista ordenada
Entrada/saída	
virtual Save	Salva os dados na lista de arquivos
virtual Load	Carrega os dados a partir da lista de arquivos
virtual Type	Obtém a lista do tipo identificada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

FreeMode

Obtém o flag de gerenciamento de memória ao excluir itens da lista.

```
bool  FreeMode() const
```

Valor do Retorno

Flag de gerenciamento de memória .

Exemplo

```
//--- example for CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get free mode flag
   bool list_free_mode=list.FreeMode();
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

FreeMode

Define o flag de gerenciamento de memória ao excluir itens da lista.

```
void  FreeMode(
   bool  mode       // New value
   )
```

Parâmetros

mode

[in] O novo valor do flag de gerenciamento de memória.

Observação

Definir o flag de gerenciamento de memória - um ponto importante no uso da classe CList. Desde que os elementos da lista são ponteiros para objetos dinâmicos, é importante determinar o que fazer com eles quando remover a partir da lista. Se o flag estiver definido, então remove um item da lista, o item é excluído automaticamente pelo operador "delete". Se o flag não está definido, é assumido que um ponteiro para o objeto excluído ainda está em algum lugar no programa do usuário e será aliviado dele (do programa) então.

Se o usuário resetar o flag de gerenciamento de memória, deve compreender a sua responsabilidade para a remoção dos itens na lista antes da conclusão do programa, porque de outra forma, a memória liberada não é ocupada pelos elementos quando eles criam um novo operador. Se o usuário não resetar o flag de gerenciamento de memória, há um outro "recife", ou seja, grandes quantidades de dados eventualmente poderiam até mesmo quebrar o seu terminal.

Usando ponteiros, itens da lista armazenados em algum lugar das variáveis locais, após a remoção da lista, irá levar a um erro crítico e trava o programa do usuário. Por padrão, o flag de gerenciamento de memória está definido, ou seja, a classe da lista é responsável por liberar os elementos de memória.

Exemplo

```
//--- example for CList::FreeMode(bool)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- reset free mode flag
   list.FreeMode(false);
   //--- use list
   //--- . . .
   //--- delete list
```

© 2000-2024, MetaQuotes Ltd.

```
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Total

Obtém o número de elementos na lista.

```
int  Total() const
```

Valor do Retorno

Número de elementos na lista.

Exemplo

```
//--- example for CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check total
   int total=list.Total();
   //--- use list
   //--- ...
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

IsSorted

Obtém a lista de flag ordenado.

```
bool  IsSorted(
   int  mode=0       // Sorting mode
   ) const
```

Parâmetros

mode=0

[in] Versão de testes para ordenação

Valor do Retorno

Flag da lista ordenada. Se a lista é ordenada pela opção determinada, então é verdadeiro, caso contrário será falso.

Observação

Flag da lista ordenada não pode ser alterado diretamente. Flag definido pelo Sort (int) e reseta qualquer método para adicionar/inserir.

Exemplo

```
//--- example for CList::IsSorted()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check sorted
   if(list.IsSorted(0))
     {
      //--- use methods for sorted list
      //--- ...
     }
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

SortMode

Obtém a versão de ordenação.

```
int  SortMode() const
```

Valor do Retorno

Opção de ordenação, ou -1 se a lista não é ordenada.

Exemplo

```
//--- example for CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- check sort mode
   int sort_mode=list.SortMode();
   //--- use list
   //--- ...
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateElement

Cria um novo item à lista.

```
CObject*  CreateElement()
```

Valor do Retorno

Se com sucesso, retorna ponteiro para o elemento recém criado. Retorna NULL se não for possível criar o elemento.

Observação

Método CreateElement () na classe CList sempre retorna NULL e não executa nenhuma ação. Se necessário, em uma classe derivada, o método CreateElement () deve ser implementado.

Exemplo

```
//--- example for CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- fill list
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Element create error");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento para o final da lista.

```
int  Add(
   CObject*  element      // Element to add
   )
```

Parâmetros

element

[in] O valor do elemento para adicionar à lista.

Valor do Retorno

Se for bem sucedido, retorna o índice do elemento adicionado, ou -1 em caso de erro.

Observação

O elemento não é adicionado à lista caso o parâmetro não passe ponteiro válido (isto é, NULL).

Exemplo

```
//--- example for CList::Add(Cobject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add 100 elements
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Element addition error");
         delete list;
         return;
        }
     }
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Insert

Elemento de inserções na lista na posição determinada.

```
int  Insert(
   CObject*  element,      // Element to insert
   int       pos           // Position
   )
```

Parâmetros

element

[in] o valor do elemento a inserir na lista

pos

[in] Posição na lista para inserir

Valor do Retorno

Se com sucesso, retorna o índice do elemento inserido, ou -1 em caso de erro.

Observação

O elemento não é adicionado à lista caso o parâmetro não passe ponteiro válido (isto é, NULL).

Exemplo

```
//--- example for CList::Insert(CObject*,int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- insert 100 elements
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Element insert error");
         delete list;
         return;
        }
     }
   //--- use list
   //--- . . .
   //--- delete list
```

© 2000-2024, MetaQuotes Ltd.

```
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

DetachCurrent

Remove um item a partir da posição atual da lista sem excluir o item "fisicamente".

```
CObject*  DetachCurrent()
```

Valor do Retorno

Ponteiro para a remoção de elementos em caso de sucesso. Retorna NULL se não pode remover o elemento.

Observação

Quando removido da lista, o item não é removido em qualquer estado do flag de gerenciamento de memória. O Ponteiro retira o elemento de liberação da lista de ingredientes após o uso.

Exemplo

```
//--- example for CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Detach error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- delete element
   delete object;
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteCurrent

Remove o elemento a partir da posição atual na lista.

```
bool  DeleteCurrent()
```

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Observação

Se ativado o gerenciamento de memória e a memória, então a remoção do elemento é liberada.

Exemplo

```
//--- example for CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove o elemento a partir da posição dada na lista.

```
bool  Delete(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] posição remove o elemento na lista.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode remover o elemento.

Observação

Se ativado o gerenciamento de memória e a memória, então a remoção do elemento é liberada.

Exemplo

```
//--- example for CList::Delete(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Delete error");
      delete list;
      return;
     }
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da lista.

```
void  Clear()
```

Observação

Se ativado o gerenciamento de memória, a memória e os itens excluídos estarão isentos.

Exemplo

```
//--- example for CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   //--- clear list
   list.Clear();
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

IndexOf

Obtém o índice do item da lista.

```
int  IndexOf(
   CObject*  element      // Pointer to the element
   )
```

Parâmetros

element

[in] Ponteiro para o item da lista.

Valor do Retorno

Item de índice na lista, ou -1.

Exemplo

```
//--- example for CList::IndexOf(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Element create error");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetNodeAtIndex

Obtém um item com o índice determinado da lista.

```
CObject*  GetNodeAtIndex(
   int  pos      // position
   )
```

Parâmetros

pos

[in] índice do item na lista.

Valor de retorno

Ponteiro para o item em caso de sucesso. NULL, se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetNodeAtIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetFirstNode

Obtém o primeiro elemento da lista.

```
CObject*  GetFirstNode()
```

Valor do Retorno

Ponteiro para o primeiro item em caso de sucesso. NULL se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetPrevNode

Obtém o elemento anterior da lista.

```
CObject*  GetPrevNode()
```

Valor do Retorno

Ponteiro para o elemento anterior em caso de sucesso. NULL, se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetCurrentNode

Obtém o item atual da lista.

```
CObject*  GetCurrentNode()
```

Valor do Retorno

Se com sucesso, retorna o ponteiro para o item atual. Retorna NULL se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetNextNode

Obtém o próximo item na lista.

```
CObject*  GetNextNode()
```

Valor do Retorno

Se com sucesso, retorna o ponteiro para o próximo item. Retorna NULL se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

GetLastNode

Obtém o último elemento da lista.

```
CObject*  GetLastNode()
```

Valor do Retorno

Ponteiro para o último elemento em caso de sucesso. NULL se não pode obter um ponteiro.

Exemplo

```
//--- example for CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add list elements
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Get node error");
      delete list;
      return;
     }
   //--- use element
   //--- . . .
   //--- do not delete element
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Sort

Ordena uma lista.

```
void  Sort(
   int  mode      // Sorting mode
   )
```

Parâmetros

mode

[in] Modo de ordenação.

Valor do Retorno

Não tem.

Observação

A ordenação da lista é sempre em ordem crescente.

Exemplo

```
//--- example for CList::Sort(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- sorting by mode 0
   list.Sort(0);
   //--- use list
   //--- ...
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

MoveToIndex

Move o item da lista atual para a posição determinada.

```
bool  MoveToIndex(
   int  pos      // Position
   )
```

Parâmetros

pos

[in] Posição na lista para mover.

Valor do Retorno

verdadeiro se com sucesso, falso - não pode mover o item.

Exemplo

```
//--- example for CList::MoveToIndex(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- move current node to begin
   list.MoveToIndex(0);
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Exchange

Muda elementos dos lugares da lista.

```
bool  Exchange(
   CObject*  node1,      // List item
   CObject*  node2       // List item
   )
```

Parâmetros

node1

[in] Lista de itens

node2

[in] Lista de itens

Valor do Retorno

verdadeiro se com sucesso. Falso se não pode mudar os elementos em alguns lugares.

Exemplo

```
//--- example for CList::Exchange(CObject*,CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- exchange
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- use list
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CompareList

Compara a lista com outra lista.

```
bool  CompareList(
   CList*  list      // With whom we compare
   )
```

Parâmetros

list

[in] Um ponteiro para uma instância de classe CList - elementos fontes para comparação.

Valor do Retorno

verdadeiro se as listas são iguais, falso - se não.

Exemplo

```
//--- example for CList::CompareList(const CList*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- create source list
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Object create error");
      delete list;
      return;
     }
   //--- fill source list
   //--- . . .
   //--- compare with another list
   bool result=list.CompareList(src);
   //--- delete lists
   delete src;
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Pesquisa

Procura por um elemento igual ao modelo na lista ordenada.

```
CObject*  Search(
   CObject*  element      // Sample
   )
```

Parâmetros

element

[in] Amostra de célula para procurar na lista.

Valor do Retorno

Ponteiro para o item encontrado em caso de sucesso. NULL, se o item não foi encontrado.

Exemplo

```
//--- example for CList::Search(CObject*)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add lists elements
   //--- . . .
   //--- sort list
   list.Sort(0);
   //--- create sample
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Sample create error");
      delete list;
      return;
     }
   //--- set sample attributes
   //--- . . .
   //--- search element
   if(list.Search(sample)!=NULL) printf("Element found");
   else                          printf("Element not found");
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva os dados na lista de arquivos.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador previamente aberto usando a função do arquivo FileOpen (...).

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CList::Save(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- add lists elements
   //--- . . .
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- delete list
   delete list;
```

© 2000-2024, MetaQuotes Ltd.

```
}
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega a lista de dados de um arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador aberto anteriormente, com a função FileOpen (...), binária

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Observação

Quando lê a partir dos itens da lista de arquivo para criar, cada elemento do método é chamado CList :: CreateElement ().

Exemplo

```
//--- example for CLoad::Load(int)
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Object create error");
      return;
     }
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   //--- use list elements
   //--- . . .
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém a lista do tipo identificada.

```
virtual int  Type()
```

Valor do Retorno

Lista do tipo identificada (para CList - 7779).

Exemplo

```
//--- example for CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Object create error");
      return;
     }
   //--- get list type
   int type=list.Type();
   //--- delete list
   delete list;
  }
```

© 2000-2024, MetaQuotes Ltd.

CTreeNode

CTreeNode é uma classe de nó da árvore binária CTree.

Descrição

Classe CTreeNode possibilita trabalhar com os nós da árvore binária Ctree. Opções de navegação através da árvore é implementado na classe. Também os métodos de trabalho com um arquivo são implementados.

Declaração

```
class CTreeNode : public CObject
```

Título

```
#include <Arrays\TreeNode.mqh>
```

Hierarquia de herança

CObject

CTreeNode

Descendentes diretos

CTree

Métodos de classe

Atributos	
Owner	Retorna/define o ponteiro do nó proprietário
Left	Retorna/define o ponteiro do nó esquerdo
Right	Retorna/define o ponteiro do nó direito
Balance	Obtém o equilíbrio do nó
BalanceL	Obtém o equilíbrio do sub-ramo esquerdo do nó
BalanceR	Obtém o equilíbrio da sub-ramo direito do nó
Criação de um novo elemento	
CreateSample	Cria uma nova instância de nó
Comparação	
RefreshBalance	Recalcula o equilíbrio do nó
Pesquisa	
GetNext	Obtém o ponteiro do próximo nó
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
SaveNode	Salva os dados do nó para um arquivo
LoadNode	Transfere os dados do nó de um arquivo
virtual Type	Obtém o identificador do tipo de nó

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Compare

Os descendentes das árvores da classe CTreeNode obtêm aplicação prática.

Um descendente de classe CTreeNode deve ter métodos predefinidos: CreateSample que cria uma nova instância de descendente da classe CTreeNode, Compare que compara os valores dos campos-chave do descendente da classe CTreeNode, Type (Se for necessário identificar um nó), SaveNode e LoadNode (Se for necessário trabalhar com um arquivo).

Vamos considerar e exemplo de um descendente da classe Ctree.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                         https://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| Describe classderived from CTreeNode.                            |
//+------------------------------------------------------------------+
//| Class CMyTreeNode.                                               |
//| Purpose: Class of element of a binary tree.                      |
//|             Descendant of class CTreeNode.                       |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- user's data
   long              m_long;           // key field of type long
   double            m_double;         // custom variable of type double
   string            m_string;         // custom variable of type string
   datetime          m_datetime;       // custom variable of type datetime

public:
                     CMyTreeNode();
   //--- methods of accessing these user's data
   long              GetLong(void)                 { return(m_long); }
   void              SetLong(long value)           { m_long=value;  }
```

```
   double            GetDouble(void)              { return(m_double); }
   void              SetDouble(double value)      { m_double=value;  }
   string            GetString(void)              { return(m_string); }
   void              SetString(string value)      { m_string=value;  }
   datetime          GetDateTime(void)            { return(m_datetime); }
   void              SetDateTime(datetime value)  { m_datetime=value;  }
   //--- methods of working with files
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- methods of creating class instances
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| CMyTreeNode class constructor.                                   |
//| INPUT:  none.                                                    |
//| OUTPUT: none.                                                    |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- initialization of user's data
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Comparison with another three node by the specified algorithm.   |
//| INPUT:  node - array element to compare,                         |
//|         mode - identifier of comparison algorithm.               |
//| OUTPUT: result of comparison (>0,0,<0).                          |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- parameter mode is ignored, because tree construction algorithm is the only one
   int res=0;
//--- explicit type casting
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Creation of a new class instance.                                |
//| INPUT:  none.                                                    |
//| OUTPUT: pointer to a new instance of class CMyTreeNode.          |
```

© 2000-2024, MetaQuotes Ltd.

```
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Write tree node data to a file.                                  |
//| INPUT:  file_handle -handle of a file pre-opened for writing.    |
//| OUTPUT: true if OK, otherwise false.                             |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- checks
   if(file_handle<0) return(false);
//--- writing user data
//--- writing custom variable of type long
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- writing custom variable of type double
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- writing custom variable of type string
   len=StringLen(m_string);
//--- write string length
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- write the string
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- writing custom variable of type datetime
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Read tree node data from a file.                                 |
//| INPUT:  file_handle -handle of a file pre-opened for reading.    |
//| OUTPUT: true if OK, otherwise false.                             |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- checks
   if(file_handle<0) return(false);
//--- reading
   if(FileIsEnding(file_handle)) return(false);
//--- reading custom variable of type char
```

© 2000-2024, MetaQuotes Ltd.

```
//--- reading custom variable of type long
   m_long=FileReadLong(file_handle);
//--- reading custom variable of type double
   m_double=FileReadDouble(file_handle);
//--- reading custom variable of type string
//--- read the string length
   len=FileReadInteger(file_handle,INT_VALUE);
//--- read the string
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- reading custom variable of type datetime
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Owner

Obtém o ponteiro do nó proprietário.

```
CTreeNode*  Owner()
```

Valor do Retorno

Ponteiro do nó-proprietário.

Owner

Define o ponteiro do nó proprietário.

```
void  Owner(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] O novo valor do ponteiro do nó proprietário.

Valor do Retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Left

Obtém o ponteiro do nó esquerdo.

```
CTreeNode*  Left()
```

Valor do Retorno

Ponteiro do nó esquerdo.

Left

Define o ponteiro do nó esquerdo.

```
void  Left(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] O novo valor do ponteiro do nó esquerdo.

Valor do Retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Right

Obtém o ponteiro do nó direito.

```
CTreeNode*  Right()
```

Valor do Retorno

O ponteiro do nó direito.

Right

Define o ponteiro do nó direito.

```
void  Right(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] O novo valor do ponteiro do nó direito.

Valor do Retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Balance

Obtém o equilíbrio do nó.

```
int  Balance() const
```

Valor do Retorno

Equilíbrio do nó.

© 2000-2024, MetaQuotes Ltd.

BalanceL

Obtém o equilíbrio do sub-ramo esquerdo do nó.

```
int  BalanceL() const
```

Valor do Retorno

Equilíbrio do sub-ramo esquerdo do nó.

© 2000-2024, MetaQuotes Ltd.

BalanceR

Obtém o equilíbrio da sub-ramo direito do nó.

```
int  BalanceR() const
```

Valor do Retorno

Equilíbrio do sub-ramo direito do nó.

© 2000-2024, MetaQuotes Ltd.

CreateSample

Cria uma nova amostra do nó.

```
virtual CTreeNode*  CreateSample()
```

Valor do Retorno

Ponteiro para a nova amostra do nó ou NULL.

© 2000-2024, MetaQuotes Ltd.

RefreshBalance

Recalcula o equilíbrio do nó.

```
int  RefreshBalance()
```

Valor do Retorno

Equilíbrio do nó.

© 2000-2024, MetaQuotes Ltd.

GetNext

Obtém o ponteiro do próximo nó.

```
CTreeNode*  GetNext(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] Nó do início da busca.

Valor do Retorno

Ponteiro do próximo nó.

© 2000-2024, MetaQuotes Ltd.

SaveNode

Escreve dados do nó num arquivo.

```
bool  SaveNode(
   int  file_handle      // handle
   )
```

Parâmetros

file_handle

[in] Manipulador de um arquivo binário que foi aberto anteriormente para escrever.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

LoadNode

Lê dados do nó de um arquivo.

```
bool  LoadNode(
   int         file_handle,      // handle
   CTreeNode*  main              // node
   )
```

Parâmetros

file_handle

[in] Manipulador de um arquivo binário que foi aberto anteriormente para leitura.

main

[in] Nó para dados.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo do nó.

```
virtual int  Type() const
```

Valor do Retorno

Identificador do tipo do nó.

© 2000-2024, MetaQuotes Ltd.

CTree

Classe CTree é uma classe da árvore binária das amostras de classe CTreeNode e seus descendentes.

Descrição

Classe CTree possibilita o trabalho com uma árvore binária de classe CTreeNode e seus descendentes. Opções de adicionar/inserir/excluir elementos e pesquisar numa árvore são implementados na classe. Além disso, métodos de trabalho com um arquivo são implementados.

Note que o mecanismo de gerenciamento de memória dinâmica não está implementado na classe CTree (como distinto das classe CList e CArrayObj). Todos nós da árvore são excluídos com a liberação de memória.

Declaração

```
class CTree : public CTreeNode
```

Título

```
#include <Arrays\Tree.mqh>
```

Hierarquia de herança

CObject
  CTreeNode
    CTree

Métodos de classe

Atributos	
Root	Obtém um nó da raiz da árvore
Criação de um novo elemento	
CreateElement	Cria uma nova instância de nó
Preenchimento	
Insert	Adiciona um nó a uma árvore
Eliminação	
Detach	Destaca um nó determinado numa árvore
Delete	Exclui um nó determinado numa árvore
Clear	Excluí todos os nós da árvore
Pesquisa	
Find	Procura por um nó numa árvore pela amostra
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
virtual Save	Salva todos os dados da árvore para um arquivo
virtual Load	Downloads de dados da árvore de um arquivo
virtual Type	Obtém identificador do tipo de árvore

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Árvores da descendência da classe CTreeNode - descendentes de classe CTree obtêm a aplicação prática.

Descendente de classe CTree deve ter um método predefinido CreateElement que cria uma nova amostra de classe descendente CTreeNode.

Vamos considerar um exemplo da descendência da classe CTree.

```
//+------------------------------------------------------------------+
//|                                                       MyTree.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| Describe class CMyTree derived from CTree.                       |
//+------------------------------------------------------------------+
//| Class CMyTree.                                                   |
//| Purpose: Construction and navigation of a binary search tree.    |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- methods of search on the tree by custom data
   CMyTreeNode*        FindByLong(long find_long);
   //--- method of creation of the tree element
   virtual CTreeNode *CreateElement();
  };
//---
```

```
CMyTree MyTree;
//+------------------------------------------------------------------+
//| Creation of a new tree node.                                     |
//| INPUT:  none.                                                    |
//| OUTPUT: pointer to the new tree node of OK, or NULL.             |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| Search of element in a list by value m_long.                     |
//| INPUT:  find_long - searched value.                              |
//| OUTPUT: pointer of a found list element, or NULL.                |
//| REMARK: none.                                                    |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- create a tree node to pass the search parameter
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "testing of class CMyTree"                                |
//+------------------------------------------------------------------+
//---  array for string initialization
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- temporary pointer to the sample of class CMyTreeNode
//---
   printf("Start test %s.",__FILE__);
//--- Fill out MyTree with samples of class MyTreeNode in the amount of extCountedNode
   beg_time=GetTickCount();
```

```
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
      if(node==NULL)
        {
         //--- emergency exit
         printf("%s (%4d): create error",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Filling time of MyTree is %d ms.",end_time-beg_time);
//--- Create a temporary tree TmpMyTree.
   CMyTree TmpMyTree;
//--- Detach 50% of tree elements (all even)
//--- and add them to the temporary tree TmpMyTree.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("Deletion time of %d elements from MyTree is %d ms.",extCountedNodes/2,end_t
//--- Return the detached
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Check work of method Save(int file_handle);
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- error writing to a file
         //--- emergency exit
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- close file before leaving!!!
         FileClose(file_handle);
         return(__LINE__);
        }
```

© 2000-2024, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
//--- Check work of method Load(int file_handle);
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- error reading from file
         //--- emergency exit
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- close file before leaving!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("End test %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Function of output of node contents to journal                   |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
               node.GetLong(),node.GetDouble(),
               node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Function of "filling" of node with random values                 |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
```

© 2000-2024, MetaQuotes Ltd.

```
   node.SetString(str_array[mode%10]);
   node.SetDateTime(10000*mode);
  }
```

© 2000-2024, MetaQuotes Ltd.

Root

Obtém o nó raiz da árvore.

```
CTreeNode*  Root() const
```

Valor do Retorno

Ponteiro do nó raiz da árvore.

© 2000-2024, MetaQuotes Ltd.

CreateElement

Cria uma nova instância do nó.

```
virtual CTreeNode*  CreateElement()
```

Valor do Retorno

Ponteiro da nova instância do nó ou NULL.

© 2000-2024, MetaQuotes Ltd.

Insert

Adiciona um nó a uma árvore.

```
CTreeNode*  Insert(
   CTreeNode*  new_node       // node
   )
```

Parâmetros

new_node

[in] ponteiro de um nó para inserir numa árvore.

Valor do Retorno

Ponteiro do nó proprietário ou NULL.

© 2000-2024, MetaQuotes Ltd.

Detach

Destaca um nó determinado a partir de uma árvore.

```
bool  Detach(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] Ponteiro do nó para destacar.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

Observação

Após destacar, o ponteiro do nó não é liberado. A árvore fica equilibrada.

© 2000-2024, MetaQuotes Ltd.

Delete

Exclui um nó determinado de uma árvore.

```
bool  Delete(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] ponteiro do nó a ser excluído.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

Observação

Após a exclusão, um ponteiro de nó é liberado. A árvore fica equilibrada.

© 2000-2024, MetaQuotes Ltd.

Clear

Exclui todos os nós de uma árvore.

```
void  Clear()
```

Valor do Retorno

Nenhum.

Observação

Após a exclusão, ponteiros de nó são liberados.

© 2000-2024, MetaQuotes Ltd.

Find

Procura por um nó numa árvore por exemplo.

```
CTreeNode*  Find(
   CTreeNode*  node       // node
   )
```

Parâmetros

node

[in] Nó que contém amostra de busca de dados.

Valor do Retorno

Ponteiro do nó encontrado ou NULL.

© 2000-2024, MetaQuotes Ltd.

Save

Grava dados da árvore num arquivo.

```
virtual bool  Save(
   int  file_handle      // handle
   )
```

Parâmetros

file_handle

[in] Manipulador de um arquivo binário que foi aberto anteriormente para escrever.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Load

Lê os dados de árvore num arquivo.

```
virtual bool  Load(
   int  file_handle      // handle
   )
```

Parâmetros

file_handle

[in] Manipulador de um arquivo binário que foi aberto anteriormente para leitura.

Valor do Retorno

verdadeiro em caso de sucesso, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador do tipo de árvore.

```
virtual int  Type() const
```

Valor do Retorno

Identificador do tipo de árvore.

© 2000-2024, MetaQuotes Ltd.

Coleções de dados genéricas

A biblioteca contém classes e interfaces para a definição de coleções genéricas, que, por sua vez, permitem que o usuário crie uma coleção fortemente tipada. Elas proporcionam uma maior comodidade e desempenho de manipulação de dados do que as coleções tipadas convencionais.

A biblioteca está localizada no diretório de trabalho do terminal na pasta Include\Generic.

Objetos:

Objeto	Descrição	Tipo
ICollection	Interface para realização de coleções de dados genéricas	INTERFACE
IEqualityComparable	Interface para realização de objetos que podem ser comparados entre si	INTERFACE
IComparable	Interface para realização de objetos que podem ser comparados entre si por meio da relação "maior, menor ou igual"	INTERFACE
IComparer	Interface para realização de uma classe universal que compara dois objetos do tipo T no que diz respeito à relação "maior, menor ou igual"	INTERFACE
IEqualityComparer	Interface para realização de uma classe universal que compara dois objetos do tipo T no que diz respeito à igualdade	INTERFACE
IList	Interface para realização de listas de dados genéricas	INTERFACE
IMap	Interface para realização de coleções genéricas do par chave-valor	INTERFACE
ISet	Interface para realização de conjuntos de dados genéricos	INTERFACE
CDefaultComparer	Classe auxiliar que implementa a interface genérica IComparer<T> com base nos métodos globais Compare	CLASS
CDefaultEqualityComparer	Classe auxiliar que implementa a interface genérica IEqualityComparer<T> usando os métodos globais Equals<T> e GetHashCode	CLASS
CArrayList	Classe genérica que implementa a interface IList<T>	CLASS

© 2000-2024, MetaQuotes Ltd.

Objeto	Descrição	Tipo
CKeyValuePair	A classe implementa o par chave-valor	CLASS
CHashMap	Classe genérica que implementa a interface IMap<TKey, TValue>	CLASS
CHashSet	Classe genérica que implementa a interface ISet<T>	CLASS
CLinkedListNode	Classe auxiliar necessária para implementar a classe CLinkedListNode<T>	CLASS
CLinkedList	Classe genérica que implementa a interface ICollection<T>	CLASS
CQueue	Classe genérica que implementa a interface ICollection<T>	CLASS
CRedBlackTreeNode	Classe auxiliar necessária para implementar a classe CRedBlackTree<T>	CLASS
CRedBlackTree	Classe genérica que implementa a interface ICollection<T>	CLASS
CSortedMap	Classe genérica que implementa a interface IMap<TKey, TValue>	CLASS
CSortedSet	Classe genérica que implementa a interface ISet<T>	CLASS
CStack	Classe genérica que implementa a interface ICollection<T>	CLASS

Métodos globais:

Método	Descrição
ArrayBinarySearch	Procura o valor especificado numa matriz unidimensional classificada em ordem ascendente usando a interface IComparable<T> para comparação de elementos
ArrayIndexOf	Procura a primeira ocorrência de um valor numa matriz unidimensional
ArrayLastIndexOf	Procura a última ocorrência de um valor numa matriz unidimensional
ArrayReverse	Altera o sequenciamento dos elementos numa matriz unidimensional
Compare	Compara dois valores no que diz respeito à relação "maior, menor ou igual"

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
Equals	Compara se dois valores são iguais
GetHashCode	Calcula o valor de código hash

© 2000-2024, MetaQuotes Ltd.

ICollection<T>

ICollection<T> é uma interface para realização de coleções de dados genéricas.

Descrição

A interface ICollection<T> define os métodos principais para trabalhar com coleções, isto é, contagem de elementos, limpeza de coleção, adição, exclusão e remoção de elementos, e outros.

Declaração

```
template<typename T>
interface ICollection
```

Cabeçalho

```
#include <Generic\Interfaces\ICollection.mqh>
```

Hierarquia de herança

ICollection

Descendentes diretos

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Métodos de classe

Método	Descrição
Add	Adiciona um elemento à coleção
Count	Retorna o número de elementos na coleção
Contains	Determina se a coleção contém um elemento com o valor especificado
CopyTo	Copia todos os elementos da coleção para a matriz especificada, a partir do índice definido
Clear	Remove todos os elementos da coleção
Remove	Remove a primeira ocorrência do elemento especificado da coleção

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à coleção.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na coleção.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a coleção contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a coleção contém um elemento com o valor especificado, caso contrário false..

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da coleção para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados os elementos da coleção.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da coleção.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do elemento especificado da coleção.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IEqualityComparable<T>

Interface IEqualityComparable<T> é uma interface para realização de objetos que podem ser comparados entre si.

Descrição

Interface IEqualityComparable<T> define métodos para obter o código hash do objeto atual e sua comparação de igualdade com outro objeto do mesmo tipo.

Declaração

```
template<typename T>
interface IEqualityComparable
```

Cabeçalho

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Hierarquia de herança

IEqualityComparable

Descendentes diretos

IComparable

Métodos de classe

Método	Descrição
Equals	Compara o objeto atual com o valor especificado
HashCode	Calcula o valor de código hash para o objeto atual

© 2000-2024, MetaQuotes Ltd.

Equals

Compara o objeto atual com o valor especificado.

```
bool Equals(
   T  value     // valor para comparação
   );
```

Parâmetros

value

[in] Valor com o qual é comparado o objeto atual.

Valor de retorno

Retorna true, se os objetos são iguais, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcula o valor de código hash para o objeto atual.

```
int HashCode();
```

Valor de retorno

Retorna o código hash.

© 2000-2024, MetaQuotes Ltd.

IComparable<T>

Interface IComparable<T> é uma interface para realização de objetos que podem ser comparados entre si por meio da relação "maior, menor ou igual".

Descrição

Interface IComparable<T> especifica o método de comparação do objeto atual com outro do mesmo tipo com base no qual a coleção destes objetos pode ser classificada.

Declaração

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Cabeçalho

```
#include <Generic\Interfaces\IComparable.mqh>
```

Hierarquia de herança

IEqualityComparable

IComparable

Descendentes diretos

CKeyValuePair

Métodos de classe

Método	Descrição
Compare	Compara o objeto atual com o valor especificado

© 2000-2024, MetaQuotes Ltd.

Compare

Compara o objeto atual com o valor especificado.

```
int Compare(
   T  value     // valor para comparação
   );
```

Parâmetros

value

[in] Valor com o qual é comparado o objeto atual.

Valor de retorno

Retorna o número que expressa a relação entre o objeto atual o objeto enviado:

- resultado inferior a zero indica que o objeto atual é menor do que o objeto enviado
- resultado igual a zero indica que o objeto atual é igual ao objeto enviado
- resultado superior a zero indica que o objeto atual é maior do que o objeto enviado

© 2000-2024, MetaQuotes Ltd.

IComparer<T>

Interface IComparer<T> é uma interface para implementar uma classe genérica que compara dois objetos do tipo T por uma relação "maior, menor ou igual".

Descrição

Interface IComparer<T> define o método de comparação de dois objetos do tipo T, com base no qual a coleção destes objetos pode ser classificada.

Declaração

```
template<typename T>
interface IComparer
```

Cabeçalho

```
#include <Generic\Interfaces\IComparer.mqh>
```

Hierarquia de herança

IComparer

Descendentes diretos

CDefaultComparer

Métodos de classe

Método	Descrição
Compare	Compara dois valores de tipo T

© 2000-2024, MetaQuotes Ltd.

Compare

Compara dois valores de tipo T.

```
int Compare(
   T  x,      // primeiro valor
   T  y       // segundo valor
   );
```

Parâmetros

x

[in] Primeiro valor para comparação.

y

[in] Primeiro valor para comparação.

Valor de retorno

Retorna o número que expressa a razão entre os dois valores comparados:

- resultado inferior a zero — x é menor que y (x<y)
- resultado igual a zero — x igual a y (x=y)
- resultado superior a zero — x maior que y (x>y)

© 2000-2024, MetaQuotes Ltd.

IEqualityComparer<T>

Interface IEqualityComparer<T> é uma interface para implementar uma classe genérica que compara dois objetos do tipo T.

Descrição

Interface IEqualityComparer<T> define métodos para obter o código hash do objeto do tipo T e a comparação de igualdade de dois objetos do tipo T.

Declaração

```
template<typename T>
interface IEqualityComparer
```

Cabeçalho

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Hierarquia de herança

IEqualityComparer

Descendentes diretos

CDefaultEqualityComparer

Métodos de classe

Método	Descrição
Equals	Compara dois valores de tipo T
HashCode	Calcula o valor de código hash a partir do objeto de tipo T

© 2000-2024, MetaQuotes Ltd.

Equals

Compara dois valores de tipo T.

```
bool Equals(
   T  x,      // primeiro valor
   T  y       // segundo valor
   );
```

Parâmetros

x

[in] Primeiro valor para comparação.

y

[in] Segundo valor para comparação.

Valor de retorno

Retorna true, se os valores são iguais, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcula o valor de código hash a partir do objeto de tipo T.

```
int HashCode(
   T  value     // objeto para cálculo
   );
```

Parâmetros

value

[in] Objeto para o qual é necessário obter o código hash.

Valor de retorno

Retorna o código hash.

© 2000-2024, MetaQuotes Ltd.

IList<T>

Interface IList<T> é uma interface para implementar listas genéricas de dados.

Descrição

Interface IList<T> define os métodos principais para trabalhar com listas, isto é, o acesso ao elemento por índice, a pesquisa e exclusão do elemento, a classificação e outros.

Declaração

```
template<typename T>
interface IList : public ICollection<T>
```

Cabeçalho

```
#include <Generic\Interfaces\IList.mqh>
```

Hierarquia de herança

ICollection

IList

Descendentes diretos

CArrayList

Métodos de classe

Método	Descrição
TryGetValue	Obtém o elemento da lista segundo o índice especificado
TrySetValue	Altera o valor da lista segundo o índice especificado
Insert	Insere um elemento na lista segundo o índice especificado
IndexOf	Procura a primeira ocorrência de um valor na lista
LastIndexOf	Procura a última ocorrência de um valor na lista
RemoveAt	Remove o elemento da lista segundo o índice especificado

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Obtém o elemento da lista segundo o índice especificado.

```
bool TryGetValue(
   const int  index,      // índice do elemento
   T&         value       // variável para gravação
   );
```

Parâmetros

index

[in] Índice de elemento da lista.

&value

[out] Variável em que será gravado o valor especificado do elemento da lista.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Altera o valor da lista segundo o índice especificado.

```
bool TrySetValue(
   const int  index,      // índice do elemento
   T          value       // novo valor
   );
```

Parâmetros

index

[in] Índice de elemento da lista.

value

[in] Novo valor que deve ser atribuído ao elemento especificado da lista.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento na lista segundo o índice especificado.

```
bool Insert(
   const int  index,      // índice para inserção
   T          item        // valor para inserção
   );
```

Parâmetros

index

[in] Índice para inserção.

item

[in] Valor que deve ser inserido no índice especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IndexOf

Procura a primeira ocorrência do valor na lista.

```
int IndexOf(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna o índice do primeiro elemento encontrado.. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

LastIndexOf

Procura a última ocorrência de um valor na lista.

```
int LastIndexOf(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna o índice do último elemento encontrado. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

RemoveAt

Remove o elemento da lista segundo o índice especificado.

```
bool RemoveAt(
   const int  index     // índice do elemento
   );
```

Parâmetros

index

[in] Índice do elemento que é necessário remover.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IMap<TKey, TValue>

Interface IMap<TKey, TValue> é uma interface para implementar coleções genéricas dos pares chave-valor.

Descrição

Interface IMap<TKey, TValue> define os métodos principais para trabalhar com coleções de dados que são armazenadas na forma de pares chave-valor.

Declaração

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Cabeçalho

```
#include <Generic\Interfaces\IMap.mqh>
```

Hierarquia de herança

ICollection

IMap

Descendentes diretos

CHashMap, CSortedMap

Métodos de classe

Método	Descrição
Add	Adiciona o par chave-valor à coleção
Contains	Determina se a coleção contém o par chave-valor com a chave e valor especificados.
Remove	Remove a primeira ocorrência do par chave-valor com a chave da coleção
TryGetValue	Obtém o elemento da coleção para a chave especificada
TrySetValue	Altera o valor do par chave-valor da coleção para a chave especificada
CopyTo	Copia todos os pares chave-valor da coleção nas matrizes especificadas, a partir do índice especificado

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona o par chave-valor à coleção.

```
bool Add(
   TKey    key,       // chave
   TValue  value      // valor
   );
```

Parâmetros

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a coleção contém o par chave-valor com a chave e valor especificados.

```
bool Contains(
   TKey    key,       // chave
   TValue  value      // valor
   );
```

Parâmetros

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true, se a coleção tem um par chave-valor para a chave e valor especificados, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do par chave-valor com a chave da coleção.

```
bool Remove(
   TKey  key     // chave
   );
```

Parâmetros

key

[in] Chave.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Obtém o elemento da coleção para a chave especificada.

```
bool TryGetValue(
   TKey     key,       // chave
   TValue&  value      // variável para gravação do valor
   );
```

Parâmetros

key

[in] Chave.

&value

[out] Variável em que será gravado o valor especificado do par chave-valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Altera o valor do par chave-valor da coleção para a chave especificada.

```
bool TrySetValue(
   TKey    key,       // chave
   TValue  value      // novo valor
   );
```

Parâmetros

key

[in] Chave.

value

[in] Novo valor que deve ser atribuído ao par chave-valor especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os pares chave-valor da coleção nas matrizes especificadas, a partir do índice especificado.

```
int CopyTo(
   TKey&      dst_keys[],       // matriz para gravação das chaves
   TValue&    dst_values[],     // matriz para gravação de valores
   const int  dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_keys[]

[out] Matriz em que serão gravados todas as chaves da coleção.

&dst_values[]

[out] Matriz em que serão gravados os valores das respectivas chaves da coleção.

dst_start=0

[in] Índice nas matrizes, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de pares chave-valor copiados.

© 2000-2024, MetaQuotes Ltd.

ISet<T>

Interface ISet<T> é uma interface para implementar conjuntos genéricos de dados.

Descrição

Interface ISet<T> define os métodos básicos para trabalhar com conjuntos, isto é, união de conjuntos e interseção de conjuntos, definição de subconjuntos estritos e não estritos e outros.

Declaração

```
template<typename T>
interface ISet : public ICollection<T>
```

Cabeçalho

```
#include <Generic\Interfaces\ISet.mqh>
```

Hierarquia de herança

ICollection

ISet

Descendentes diretos

CHashSet, CSortedSet

Métodos de classe

Método	Descrição
ExceptWith	Executa a operação de diferença da coleção atual e coleção transmitida (matriz)
IntersectWith	Executa a operação de interseção da coleção atual e coleção transmitida (matriz)
SymmetricExceptWith	Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz)
UnionWith	Executa a operação de união da coleção atual e coleção transmitida (matriz)
IsProperSubsetOf	Determina se o conjunto atual é subconjunto estrito da coleção especificada ou matriz
IsProperSupersetOf	Determina se o conjunto atual é superconjunto estrito da coleção especificada ou matriz
IsSubsetOf	Determina se o conjunto atual é subconjunto da coleção especificada ou matriz
IsSupersetOf	Determina se o conjunto atual é superconjunto da coleção especificada ou matriz

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
Overlaps	Determina se o conjunto atual se cruza com a coleção especificada ou matriz
SetEquals	Determina se o conjunto atual contém todos os elementos da coleção especificada ou matriz

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Executa a operação de diferença da coleção atual e coleção transmitida (matriz). Ou seja, remove da coleção atual (matriz) todos os elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção que será subtraída a partir do conjunto atual.

&collection[]

[in] Matriz que será subtraída a partir do conjunto atual.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Executa a operação de interseção da coleção atual e coleção transmitida (matriz). Ou seja, deixa na coleção atual apenas aqueles elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a multiplicação.

&collection[]

[in] Matriz a partir da qual será construída a multiplicação.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz). Ou seja, na coleção atual conterá apenas aqueles elementos que estavam no objeto original ou no transmitido, mas não ao mesmo tempo em ambos.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual é construída a diferença simétrica.

&collection[]

[in] Matriz a partir da qual é construída a diferença simétrica.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Executa a operação de união da coleção atual e coleção transmitida (matriz). Ou seja, adiciona à coleção atual (matriz) os elementos em falta da coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a soma.

&collection[]

[in] Matriz a partir da qual será construída a soma.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Determina se o conjunto atual é subconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Determina se o conjunto atual é superconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um superconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Determina se o conjunto atual é subconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Determina se o conjunto atual é superconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um superconjunto, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Determina se o conjunto atual se cruza com a coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para definir o cruzamento.

&collection[]

[in] Matriz para definir o cruzamento.

Valor de retorno

Retorna true, se houver uma interseção entre o conjunto atual e a coleção ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Determina se o conjunto atual contém todos os elementos da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para associação de elementos.

&collection[]

[in] Coleção para associação de elementos.

Valor de retorno

Retorna true, se, no conjunto atual, estiverem todos os elementos da coleção especificada ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CDefaultComparer<T>

Classe CDefaultComparer<T> é uma classe auxiliar que implementa a interface genérica IComparer<T> com base nos métodos globais Compare.

Descrição

Classe CDefaultComparer<T> é usada por padrão nas coleções genéricas de dados, se o usuário não usa explicitamente outra classe que implementa a interface IComparer<T>.

Declaração

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Cabeçalho

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Hierarquia de herança

IComparer
  CDefaultComparer

Métodos de classe

Método	Descrição
Compare	Compara dois valores de tipo T

© 2000-2024, MetaQuotes Ltd.

Compare

Compara dois valores de tipo T.

```
int Compare(
   T  x,      // primeiro valor
   T  y       // segundo valor
   );
```

Parâmetros

x

[in] Primeiro valor para comparação.

y

[in] Segundo valor para comparação.

Valor de retorno

Retorna o número que expressa a razão entre os dois valores comparados:

- resultado inferior a zero — x é menor que y (x<y)
- resultado igual a zero — x igual a y (x=y)
- resultado superior a zero — x maior que y (x>y)

Observação

A comparação dos valores x e y ocorre com base numa das sobrecargas do método global Compare, dependendo do tipo T.

© 2000-2024, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

Classe CDefaultEqualityComparer<T> é uma classe auxiliar que implementa a interface genérica IEqualityComparer<T> com base nos métodos globais Equals<T> e GetHashCode.

Descrição

Classe CDefaultEqualityComparer<T> é usada por padrão nas coleções genéricas de dados, se o usuário não usa explicitamente outra classe que implementa a interface IEqualityComparer<T>.

Declaração

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Cabeçalho

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Hierarquia de herança

IEqualityComparer
  CDefaultEqualityComparer

Métodos de classe

Método	Descrição
Equals	Compara dois valores de tipo T
HashCode	Calcula o valor de código hash a partir do objeto de tipo T

© 2000-2024, MetaQuotes Ltd.

Equals

Compara dois valores de tipo T.

```
bool Equals(
   T  x,      // primeiro valor
   T  y       // segundo valor
   );
```

Parâmetros

x

[in] Primeiro valor para comparação.

y

[in] Segundo valor para comparação.

Valor de retorno

Retorna true, se os valores são iguais, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcula o valor de código hash a partir do objeto de tipo T.

```
int HashCode(
   T  value     // objeto para cálculo
   );
```

Parâmetros

value

[in] Objeto para o qual é necessário obter o código hash.

Valor de retorno

Retorna o código hash.

© 2000-2024, MetaQuotes Ltd.

CRedBlackTreeNode<T>

Classe CRedBlackTree<T> é uma classe auxiliar necessária para implementar a classe CRedBlackTree<T>.

Descrição

Classe CRedBlackTreeNode<T> é o nó da árvore vermelho-preto CRedBlackTree<T>. Na classe são implementados os métodos para navegar pela árvore.

Declaração

```
template<typename T>
class CRedBlackTreeNode
```

Cabeçalho

```
#include <Generic\RedBlackTree.mqh>
```

Métodos de classe

Método	Descrição
Value	Retorna e define o valor do nó
Parent	Retorna e define o ponteiro para o nó pai
Left	Retorna e define o ponteiro para o nó esquerdo
Right	Retorna e define o ponteiro para o nó direito
Color	Retorna e define a cor do nó
IsLeaf	Determina se é um nó com folha
CreateEmptyNode	Cria um novo nó a preto sem ancestrais e descendentes, e retorna um ponteiro para ele

© 2000-2024, MetaQuotes Ltd.

Value (método Get)

Retorna o valor do nó.

```
T Value();
```

Valor de retorno

Retorna o valor do nó.

Value (método Set)

Define o valor do nó.

```
void Value(
   T  value     // valor do nó
   );
```

Parâmetros

value

[in] Valor do nó.

© 2000-2024, MetaQuotes Ltd.

Parent (método Get)

Retorna o ponteiro para o nó pai.

```
CRedBlackTreeNode<T>* Parent();
```

Valor de retorno

Retorna o ponteiro para o nó pai.

Parent (método Set)

Define o ponteiro para o nó pai.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // ponteiro para o nó pai
   );
```

Parâmetros

*node

[in] Ponteiro para o nó pai.

© 2000-2024, MetaQuotes Ltd.

Left (método Get)

Retorna o ponteiro para o nó esquerdo.

```
CRedBlackTreeNode<T>* Left();
```

Valor de retorno

Retorna o ponteiro para o nó esquerdo.

Left (método Set)

Define o ponteiro para o nó esquerdo.

```
void Left(
   CRedBlackTreeNode<T>*  node     // ponteiro para p nó esquerdo
   );
```

Parâmetros

*node

[in] Ponteiro para o nó esquerdo.

© 2000-2024, MetaQuotes Ltd.

Right (método Get)

Retorna o ponteiro para o nó direito.

```
CRedBlackTreeNode<T>* Right();
```

Valor de retorno

Retorna o ponteiro para o nó direito.

Right (método Set)

Define o ponteiro para o nó direito.

```
void Right(
   CRedBlackTreeNode<T>*  node     // ponteiro para o nó direito
   );
```

Parâmetros

*node

[in] Ponteiro para o nó direito.

© 2000-2024, MetaQuotes Ltd.

Color (método Get)

Retorna a cor do nó.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Valor de retorno

Retorna a cor do nó.

Color (método Set)

Define a cor do nó.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // cor do nó
   );
```

Parâmetros

clr

[in] Cor do nó.

Observação

Cor do nó é definida pelo valor da enumeração ENUM_RED_BLACK_TREE_NODE_TYPE e pode ser de dois tipos:

- RED_BLACK_TREE_NODE_RED — cor de nó vermelha;
- RED_BLACK_TREE_NODE_BLACK — cor de nó preta.

© 2000-2024, MetaQuotes Ltd.

IsLeaf

Determina se é um nó com folha.

```
bool IsLeaf();
```

Valor de retorno

Retorna true, se o nó é uma folha, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CreateEmptyNode

Cria um novo nó a preto sem ancestrais e descendentes, e retorna um ponteiro para ele.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Valor de retorno

Retorna o ponteiro para o novo nó.

© 2000-2024, MetaQuotes Ltd.

CLinkedListNode<T>

Classe CLinkedListNode<T> é uma classe auxiliar necessária para implementar a classe CLinkedListNode<T>.

Descrição

Classe CLinkedListNode<T> é o nó da lista bidirecional CLinkedListNode<T>. Na classe são implementados os métodos para navegar pela lista.

Declaração

```
template<typename T>
class CLinkedListNode
```

Cabeçalho

```
#include <Generic\LinkedList.mqh>
```

Métodos de classe

Método	Descrição
List	Retorna e define o ponteiro para a lista bidirecional CLinkedList<T>
Next	Retorna e define o ponteiro para o próximo nó
Previous	Retorna e define o ponteiro para o nó anterior
Value	Retorna e define o valor do nó

© 2000-2024, MetaQuotes Ltd.

List (método Get)

Retorna o ponteiro para a lista bidirecional CLinkedList<T>

```
CLinkedList<T>* List();
```

Valor de retorno

Retorna o ponteiro para a lista bidirecional CLinkedList<T>.

List (método Set)

Define o ponteiro para a lista bidirecional CLinkedList<T>.

```
void List(
   CLinkedList<T>*  value     // ponteiro para a lista
   );
```

Parâmetros

*value

[in] Ponteiro para a lista bidirecional CLinkedList<T>.

© 2000-2024, MetaQuotes Ltd.

Next (método Get)

Retorna o ponteiro para o próximo nó.

```
CLinkedListNode<T>* Next();
```

Valor de retorno

Retorna o ponteiro para o próximo nó.

Next (método Set)

Define o ponteiro para o próximo nó

```
void Next(
   CLinkedListNode<T>*  value     // ponteiro para o próximo nó
   );
```

Parâmetros

value

[in] Ponteiro para o próximo nó.

© 2000-2024, MetaQuotes Ltd.

Previous (método Get)

Retorna o ponteiro para o nó anterior.

```
CLinkedListNode<T>* Previous();
```

Valor de retorno

Retorna o ponteiro para o nó anterior.

Previous (método Set)

Define o ponteiro para o nó anterior.

```
void Previous(
   CLinkedListNode<T>*  value     // ponteiro para o nó anterior
   );
```

Parâmetros

value

[in] Ponteiro para o nó anterior.

© 2000-2024, MetaQuotes Ltd.

Value (método Get)

Retorna o valor do nó.

```
T Value();
```

Valor de retorno

Retorna o valor do nó.

Value (método Set)

Define o valor do nó.

```
void Value(
   T  value     // valor do nó
   );
```

Parâmetros

value

[in] Valor do nó.

© 2000-2024, MetaQuotes Ltd.

CKeyValuePair<TKey,TValue>

Classe CKeyValuePair<TKey, TValue> é uma classe que implementa o par chave-valor.

Descrição

Classe CKeyValuePair<TKey, TValue> implementa métodos para trabalhar diretamente como a chave e o valor do par chave-valor.

Declaração

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Cabeçalho

```
#include <Generic\HashMap.mqh>
```

Hierarquia de herança

IEqualityComparable
  IComparable
    CKeyValuePair

Métodos de classe

Método	Descrição
Key	Retorna e define a chave do par chave-valor.
Value	Retorna e define o valor do par chave-valor.
Clone	Cria um novo par chave-valor, cujos chave e valor são iguais aos atuais
Compare	Compara o par chave-valor atual com o especificado.
Equals	Compara se o par chave-valor atual é igual ao especificado.
HashCode	Calcula o valor de código hash a partir do par chave-valor

© 2000-2024, MetaQuotes Ltd.

Key (método Get)

Retorna a chave do par chave-valor.

```
TKey Key();
```

Valor de retorno

Retorna a chave.

Key (método Set)

Define o valor de chave dos pares chave-valor.

```
void Key(
   TKey  key     // chave
   );
```

Parâmetros

key

[in] Chave.

© 2000-2024, MetaQuotes Ltd.

Value (método Get)

Retorna o valor do par chave-valor.

```
TValue Value();
```

Valor de retorno

Retorna o valor.

Value (método Set)

Define o valor do par chave-valor.

```
void Value(
   TValue  value     // valor
   );
```

Parâmetros

value

[in] Valor

© 2000-2024, MetaQuotes Ltd.

Clone

Cria um novo par chave-valor, cujos chave e valor são iguais aos atuais.

```
TValue>* Clone();
```

Valor de retorno

Retorna o novo par chave-valor

© 2000-2024, MetaQuotes Ltd.

Compare

Compara o par chave-valor atual com o especificado.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // par para comparação
   );
```

Parâmetros

pair

[in] Par para comparação.

Valor de retorno

Retorna o número que expressa a relação entre o par chave-valor e o par enviado:

- resultado inferior a zero indica que o par chave-valor atual é menor do que o par enviado
- resultado igual a zero indica que o par chave-valor atual é igual ao par enviado
- resultado superior a zero indica que o par chave-valor atual é maior do que o par enviado

Observação

Comparação de dois pares chave-valor é desencadeada pela chave.

© 2000-2024, MetaQuotes Ltd.

Equals

Compara se o par chave-valor atual é igual ao especificado.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // par para comparação
   );
```

Parâmetros

pair

[in] Par para comparação

Valor de retorno

Retorna true, se os pares chave-valor são iguais, caso contrário, false.

Observação

Comparação de dois pares chave-valor é desencadeada pela chave.

© 2000-2024, MetaQuotes Ltd.

HashCode

Calcula o valor de código hash a partir do par chave-valor.

```
int HashCode();
```

Valor de retorno

Retorna o código hash.

Observação

Código hash do par chave-valor é igual ao código hash da chave.

© 2000-2024, MetaQuotes Ltd.

CArrayList<T>

Classe CArrayList<T> é uma classe genérica que implementa a interface IList<T>.

Descrição

Classe CArrayList<T> é uma implementação de uma lista dinâmica de dados de tipo T. Esta classe fornece os métodos principais para trabalhar com uma lista, isto é, o acesso ao elemento por índice, a pesquisa e exclusão do elemento, a classificação e outros.

Declaração

```
template<typename T>
class CArrayList : public IList<T>
```

Cabeçalho

```
#include <Generic\ArrayList.mqh>
```

Hierarquia de herança

ICollection
  IList
    CArrayList

Métodos de classe

Método	Descrição
Capacity	Retorna e define a capacidade atual da lista
Count	Retorna o número de elementos na lista
Contains	Determina se a lista contém um elemento com o valor especificado
TrimExcess	Reduz a capacidade de lista para o número real de elementos
TryGetValue	Obtém o elemento da lista segundo o índice especificado
TrySetValue	Define o valor do elemento da lista segundo o índice especificado
Add	Adiciona um elemento à lista
AddRange	Adiciona uma coleção ou uma matriz de elementos na lista
Insert	Insere um elemento na lista segundo o índice especificado
InsertRange	Insere uma coleção ou uma matriz de elementos na lista segundo índice especificado
CopyTo	Copia todos os elementos da lista para a matriz especificada, a partir do índice definido
BinarySearch	Procura o valor especificado numa matriz unidimensional classificada em ordem ascendente

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
IndexOf	Procura a primeira ocorrência de um valor na lista
LastIndexOf	Procura a última ocorrência de um valor na lista
Clear	Remove todos os elementos da coleção
Remove	Remove a primeira ocorrência do elemento especificado da lista
RemoveAt	Remove o elemento especificado da lista segundo o índice definido
RemoveRange	Remove um intervalo de elementos da lista
Reverse	Altera o sequenciamento dos elementos na lista
Sort	Realiza a classificação dos elementos da lista

© 2000-2024, MetaQuotes Ltd.

Capacity (método Get)

Retorna a capacidade atual da lista.

```
int Capacity();
```

Valor de retorno

Retorna a capacidade atual da lista.

Capacity (método Set)

Define a capacidade atual da lista.

```
void Capacity(
   const int  capacity     // valor de capacidade
   );
```

Parâmetros

capacity

[in] Novo valor de capacidade.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na lista.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a lista contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a lista contém um elemento com o valor especificado, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Reduz a capacidade de lista para o número real de elementos, assim, liberando a memória não utilizada.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Obtém o elemento da lista segundo o índice especificado.

```
bool TryGetValue(
   const int  index,      // índice
   T&         value       // variável para gravação
   );
```

Parâmetros

index

[in] Índice do elemento da lista, cujo valor é necessário obter.

&value

[out] Variável para gravar o valor do elemento.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Define o valor do elemento da lista segundo o índice especificado.

```
bool TrySetValue(
   const int  index,      // índice
   T          value       // valor do elemento
   );
```

Parâmetros

index

[in] Índice do elemento da lista, cujo valor é necessário definir.

value

[in] Valor a ser definido para o elemento da lista.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à lista.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AddRange

Adiciona uma coleção ou uma matriz de elementos na lista.

Versão para acrescentar uma matriz.

```
bool AddRange(
   const T&  array[]                // matriz para adicionar
   );
```

Versão para adição da coleção.

```
bool AddRange(
   ICollection<T>*  collection     // coleção para adicionar
   );
```

Parâmetros

&array[]

[in] Matriz para adição.

**collection*

[in] Coleção para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Insert

Insere um elemento na lista segundo o índice especificado.

```
bool Insert(
   const int  index,      // índice para inserção
   T          item        // valor para inserção
   );
```

Parâmetros

index

[in] Índice para inserção.

item

[in] Valor que deve ser inserido no índice especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

InsertRange

Insere uma coleção ou uma matriz de elementos na lista segundo o índice especificado.

Versão para inserção da matriz.

```
bool InsertRange(
   const int  index,              // índice para inserção
   const T&   array[]             // matriz para inserção
   );
```

Versão para inserção da coleção.

```
bool InsertRange(
   const int        index,        // índice para inserção
   ICollection<T>*  collection    // coleção para inserção
   );
```

Parâmetros

index

[in] Índice para inserção.

&array[]

[in] Matriz que deve ser inserida segundo o índice especificado.

**collection*

[in] Coleção que deve ser inserida segundo o índice especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da lista para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que são gravados os elementos da lista.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

BinarySearch

Procura o valor especificado numa matriz unidimensional classificada em ordem ascendente.

Versão para pesquisa segundo o intervalo de valores definido e com uso de uma classe que implementa a interface IComparable<T> para comparação de elementos.

```
int BinarySearch(
   const int      index,        // índice de partida
   const int      count,        // intervalo de pesquisa
   T              item,         // valor procurado
   IComparer<T>*  comparer      // interface para comparação
   );
```

Versão para pesquisa com uso de uma classe que implementa a interface IComparable<T> para comparação de elementos.

```
int BinarySearch(
   T              item,         // valor procurado
   IComparer<T>*  comparer      // interface para comparação
   );
```

Versão para pesquisa com uso do método global ::Compare para comparação de elementos.

```
int BinarySearch(
   T  item                      // valor procurado
   );
```

Parâmetros

index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

item

[in] Valor procurado.

**comparer*

[in] Interface para comparação de elementos.

Valor de retorno

Retorna o índice do elemento encontrado. Se o valor procurado não for encontrado, ele retornará o índice do elemento menor, cujo valor é mais próximo de todos.

© 2000-2024, MetaQuotes Ltd.

IndexOf

Procura a primeira ocorrência do valor na lista.

Versão para procurar na lista inteira.

```
int IndexOf(
   T  item                    // valor procurado
   );
```

Versão para pesquisa com a posição especificada, até o final da lista.

```
int IndexOf(
   T          item,           // valor procurado
   const int  start_index     // índice de partida
   );
```

Versão para pesquisa com a posição especificada, no intervalo definido.

```
int IndexOf(
   T          item,           // valor procurado
   const int  start_index,    // índice de partida
   const int  count           // intervalo de pesquisa
   );
```

Parâmetros

item

[in] Valor procurado.

start_index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

Valor de retorno

Retorna o índice do primeiro elemento encontrado.. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

LastIndexOf

Procura a última ocorrência de um valor na lista.

Versão para procurar na lista inteira.

```
int LastIndexOf(
   T  item                    // valor procurado
   );
```

Versão para pesquisa com a posição especificada, até o final da lista.

```
int LastIndexOf(
   T          item,           // valor procurado
   const int  start_index     // índice de partida
   );
```

Versão para pesquisa com a posição especificada, no intervalo definido.

```
int LastIndexOf(
   T          item,           // valor procurado
   const int  start_index,    // índice de partida
   const int  count           // intervalo de pesquisa
   );
```

Parâmetros

item

[in] Valor procurado.

start_index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

Valor de retorno

Retorna o índice do último elemento encontrado. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da coleção.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do elemento especificado da lista.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveAt

Remove o elemento especificado da lista segundo o índice definido.

```
bool RemoveAt(
   const int  index     // índice
   );
```

Parâmetros

index

[in] Índice do elemento excluído.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveRange

Remove um intervalo de elementos da lista.

```
bool RemoveRange(
   const int  start_index,      // índice de partida
   const int  count             // número de elementos
   );
```

Parâmetros

start_index

[in] Índice de partida, a partir do qual começa a remoção.

count

[in] Número de elementos a serem excluídos.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Reverse

Altera o sequenciamento dos elementos na lista.

Versão para trabalhar com a lista inteira.

```
bool Reverse();
```

Versão para trabalhar com o intervalo de elementos da lista especificado.

```
bool Reverse(
   const int  start_index,     // índice de partida
   const int  count            // número de elementos
   );
```

Parâmetros

start_index

[in] Índice inicial.

count

[in] Número de elementos da lista envolvidos na operação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Sort

Realiza a classificação dos elementos da lista.

Versão para classificação de todos os elementos da lista.

```
bool Sort();
```

Versão para classificação de todos os elementos da lista com uso de uma classe que implementa a interface IComparable<T> para comparação de elementos.

```
bool Sort(
   IComparer<T>*  comparer         // interface para comparação
   );
```

Versão para classificação do intervalo de elementos da lista com uso de uma classe que implementa a interface IComparable<T> para comparação de elementos.

```
bool Sort(
   const int      start_index,     // índice de partida
   const int      count,           // número de elementos
   IComparer<T>*  comparer         // interface para comparação
   );
```

Parâmetros

**comparer*

[in] Interface para comparação de elementos.

start_index

[in] Índice de partida, a partir do qual começa a classificação.

count

[in] Comprimento do intervalo de classificação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CHashMap<TKey,TValue>

Classe CHashMap<TKey, TValue> é uma classe genérica que implementa a interface IMap<TKey, TValue>.

Descrição

Classe CHashMap<TKey, TValue> é uma implementação de uma tabela de hash dinâmica, cujos dados são armazenados como pares chave-valor desordenados em conformidade com os requisitos de exclusividade de chave. Essa classe fornece os métodos básicos para trabalhar com uma tabela de hash, isto é, acesso ao valor da chave, procurar e eliminar par chave-valor e outros.

Declaração

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey,TValue>
```

Cabeçalho

```
#include <Generic\HashMap.mqh>
```

Hierarquia de herança

ICollection
  IMap
    CHashMap

Métodos de classe

Método	Descrição
Add	Adiciona o par chave-valor à tabela de hash
Count	Retorna o número de elementos na tabela de hash
Comparer	Retorna o ponteiro para a interface IEqualityComparer<T> que é utilizada para a classificação da tabela de hash
Contains	Determina se a lista de hash contém o par chave-valor especificado.
ContainsKey	Determina se a lista de hash contém o par chave-valor com a chave especificada.
ContainsValue	Classe CHashMap<TKey, TValue> é uma classe genérica que implementa a interface IMap<TKey, TValue>
CopyTo	Copia todos os pares chave-valor da tabela de hash nas matrizes especificadas, a partir do índice especificado
Clear	Remove todos os elementos da tabela de hash
Remove	Remove a primeira ocorrência do par chave-valor da tabela de hash
TryGetValue	Obtém o elemento da tabela de hash para a chave especificada

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
TrySetValue	Altera o valor do par chave-valor da tabela de hash para a chave especificada

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona o par chave-valor à tabela de hash.

Versão para adicionar o par chave-valor gerado.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // par chave-valor
   );
```

Versão para adicionar o novo par chave-valor com a chave e valor especificado.

```
bool Add(
   TKey    key,                         // chave
   TValue  value                        // valor
   );
```

Parâmetros

pair

[in] Par chave-valor.

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na tabela de hash.

```
int Count();
```

© 2000-2024, MetaQuotes Ltd.

Comparer

Retorna o ponteiro para a interface IEqualityComparer<T> que é utilizada para a organização da tabela de hash.

```
IEqualityComparer<TKey>* Comparer() const;
```

Valor de retorno

Retorna o ponteiro para a interface IEqualityComparer<T>.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a lista de hash contém o par chave-valor especificado.

Versão para trabalhar com o par chave-valor gerado.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // par chave-valor
   );
```

Versão para trabalhar com o par chave-valor sob a forma de uma chave e valor especificados separadamente.

```
bool Contains(
   TKey   key,                          // chave
   TValue value                         // valor
   );
```

Parâmetros

item

[in] Par chave-valor.

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true, se na tabela de hash houver um par chave-valor com a chave e valor especificados, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ContainsKey

Determina se a lista de hash contém o par chave-valor com a chave especificada.

```
bool ContainsKey(
   TKey  key     // chave
   );
```

Parâmetros

key

[in] Chave.

Valor de retorno

Retorna true, se na tabela de hash houver um par chave-valor com a chave e valor especificados, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ContainsValue

Determina se a lista de hash contém o par chave-valor com o valor especificado.

```
bool ContainsValue(
   TValue  value     // valor
   );
```

Parâmetros

value

[in] Valor.

Valor de retorno

Retorna true, se na tabela de hash houver um par chave-valor com o valor especificado, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os pares chave-valor da tabela de hash nas matrizes especificadas, a partir do índice especificado.

Versão para cópia da tabela de hash na matriz dos pares chave-valor.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // matriz para gravação dos pares cha
   const int                    dst_start=0      // índice de partida para gravação
   );
```

Versão para cópia da tabela de hash em matrizes separadas para as chaves e valores.

```
int CopyTo(
   TKey&      dst_keys[],                        // matriz para gravação de chaves
   TValue&    dst_values[],                      // matriz para gravação de valores
   const int  dst_start=0                        // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados todos os pares da tabela de hash.

&dst_keys[]

[out] Matriz em que serão gravadas todas as chaves da tabela de hash.

&dst_values[]

[out] Matriz em que serão armazenados todos os valores da tabela de hash.

dst_start=0

[in] Índice da matriz de partir da qual começa a cópia.

Valor de retorno

Retorna o número de pares chave-valor copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da tabela de hash.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do par chave-valor da tabela de hash.

Versão para remover o par chave-valor após formado o par chave-valor.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // par chave-valor
   );
```

Versão para remover o par chave-valor segundo a chave.

```
bool Remove(
   TKey  key                            // chave
   );
```

Parâmetros

*item

[in] Par chave-valor.

key

[in] Chave.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Obtém o elemento da tabela de hash para a chave especificada.

```
bool TryGetValue(
   TKey     key,       // chave
   TValue&  value      // variável para gravação do valor
   );
```

Parâmetros

key

[in] Chave.

&value

[out] Variável em que será gravado o valor especificado do par chave-valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Altera o valor do par chave-valor da tabela de hash para a chave especificada.

```
bool TrySetValue(
   TKey    key,       // chave
   TValue  value      // novo valor
   );
```

Parâmetros

key

[in] Chave.

value

[in] Novo valor que deve ser atribuído ao par chave-valor especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CHashSet<T>

Classe CHashSet<T> é uma classe genérica que implementa a interface ISet<T>.

Descrição

Classe CHashSet<T> é uma implementação do conjunto de dados dinâmico de tipo T desordenado, em conformidade com os requisitos de exclusividade de cada valor. Essa classe fornece os métodos básicos para trabalhar com conjuntos e operações sobre eles, isto é, união de conjuntos e interseção de conjuntos, definição de subconjuntos estritos e não estritos e outros.

Declaração

```
template<typename T>
class CHashSet : public ISet<T>
```

Cabeçalho

```
#include <Generic\HashSet.mqh>
```

Hierarquia de herança

ICollection
  ISet
    CHashSet

Métodos de classe

Método	Descrição
Add	Adiciona um elemento ao conjunto
Count	Retorna o número de elementos no conjunto
Comparer	Determina se o conjunto contém um elemento com o valor especificado
Contains	Retorna o ponteiro para a interface IEqualityComparer<T> que é utilizada para a organização do conjunto
TrimExcess	Reduz a capacidade do conjunto para o número real de elementos, assim, liberando a memória não utilizada
CopyTo	Copia todos os elementos do conjunto para a matriz especificada, a partir do índice especificado
Clear	Remove todos os elementos do conjunto
Remove	Remove o elemento especificado do conjunto
ExceptWith	Executa a operação de diferença da coleção atual e coleção transmitida (matriz)
IntersectWith	Executa a operação de interseção da coleção atual e coleção transmitida (matriz)

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
SymmetricExceptWith	Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz)
UnionWith	Executa a operação de união da coleção atual e coleção transmitida (matriz)
IsProperSubsetOf	Determina se o conjunto atual é subconjunto estrito da coleção especificada ou matriz
IsProperSupersetOf	Determina se o conjunto atual é superconjunto estrito da coleção especificada ou matriz
IsSubsetOf	Determina se o conjunto atual é subconjunto da coleção especificada ou matriz
IsSupersetOf	Determina se o conjunto atual é superconjunto da coleção especificada ou matriz
Overlaps	Determina se o conjunto atual se cruza com a coleção especificada ou matriz
SetEquals	Determina se o conjunto atual contém todos os elementos da coleção especificada ou matriz

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento ao conjunto.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos no conjunto.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se o conjunto contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se o conjunto contém um elemento com o valor especificado, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retorna o ponteiro para a interface IEqualityComparer<T> que é utilizada para a organização do conjunto.

```
IEqualityComparer<T>* Comparer() const;
```

Valor de retorno

Retorna o ponteiro para a interface IEqualityComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Reduz a capacidade do conjunto para o número real de elementos, assim, liberando a memória não utilizada.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos do conjunto para a matriz especificada, a partir do índice especificado.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que são gravados os elementos do conjunto.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos do conjunto.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove o elemento especificado do conjunto.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Executa a operação de diferença da coleção atual e coleção transmitida (matriz). Ou seja, remove da coleção atual (matriz) todos os elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção que será subtraída a partir do conjunto atual.

&collection[]

[in] Matriz que será subtraída a partir do conjunto atual.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Executa a operação de interseção da coleção atual e coleção transmitida (matriz). Ou seja, deixa na coleção atual apenas aqueles elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a multiplicação.

&collection[]

[in] Matriz a partir da qual será construída a multiplicação.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz). Ou seja, na coleção atual conterá apenas aqueles elementos que estavam no objeto original ou no transmitido, mas não ao mesmo tempo em ambos.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual é construída a diferença simétrica.

&collection[]

[in] Matriz a partir da qual é construída a diferença simétrica.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Executa a operação de união da coleção atual e coleção transmitida (matriz). Ou seja, adiciona à coleção atual (matriz) os elementos em falta da coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a soma.

&collection[]

[in] Matriz a partir da qual será construída a soma.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Determina se o conjunto atual é subconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Determina se o conjunto atual é superconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um superconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Determina se o conjunto atual é subconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Determina se o conjunto atual é superconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual for um superconjunto, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Determina se o conjunto atual se cruza com a coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para definir o cruzamento.

&collection[]

[in] Matriz para definir o cruzamento.

Valor de retorno

Retorna true, se houver uma interseção entre o conjunto atual e a coleção ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Determina se o conjunto atual contém todos os elementos da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para associação de elementos.

&collection[]

[in] Matriz para associação de elementos.

Valor de retorno

Retorna true, se, no conjunto atual, estiverem todos os elementos da coleção especificada ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CLinkedList<T>

Classe CLinkedList<T> é uma classe genérica que implementa a interface ICollection<T>.

Descrição

Classe CLinkedList<T> é uma implementação de uma lista dinâmica bidirecional de dados de tipo T. Esta classe fornece os métodos principais para trabalhar com uma lista bidirecional, isto é, adição, exclusão, pesquisa do elemento e outros.

Declaração

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Cabeçalho

```
#include <Generic\LinkedList.mqh>
```

Hierarquia de herança

ICollection
  CLinkedList

Métodos de classe

Método	Descrição
Add	Adiciona um elemento à lista bidirecional
AddAfter	Adiciona um elemento à lista bidirecional após o nó especificado
AddBefore	Adiciona um elemento à lista bidirecional antes do nó especificado
AddFirst	Adiciona um elemento ao topo da lista bidirecional
AddLast	Adiciona um elemento ao final da lista bidirecional
Count	Retorna o número de elementos na lista bidirecional
Head	Retorna o ponteiro para o primeiro nó na lista bidirecional
First	Retorna o ponteiro para o primeiro nó na lista bidirecional
Last	Retorna o ponteiro para o último nó na lista bidirecional
Contains	Determina se a lista bidirecional contém um elemento com o valor especificado
CopyTo	Copia todos os elementos da lista bidirecional para a matriz especificada, a partir do índice definido
Clear	Remove todos os elementos da coleção
Remove	Remove a primeira ocorrência do elemento especificado da lista bidirecional

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
RemoveFirst	Remove o primeiro elemento da lista bidirecional
RemoveLast	Remove o último elemento da lista bidirecional
Find	Procura a primeira ocorrência de um valor numa matriz bidirecional
FindLast	Procura a última ocorrência de um valor numa matriz bidirecional

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à lista bidirecional.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AddAfter

Adiciona um elemento à lista bidirecional após o nó especificado.

Versão para adição de um elemento segundo o valor.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*  node,         // nó seguido pela adição
   T                    value         // elemento a ser adicionado
   );
```

Valor de retorno

Retorna o ponteiro para o nó adicionado.

Versão para adição de um elemento, como o do nó formado, segundo o valor.

```
bool AddAfter(
   CLinkedListNode<T>*  node,         // nó seguido pela adição
   CLinkedListNode<T>*  new_node      // nó a ser adicionado
   );
```

Parâmetros

**node*

[in] Nó da lista bidirecional, após o qual será adicionado o novo elemento.

value

[in] Elemento para adição.

**new_node*

[in] Nó para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AddBefore

Adiciona um elemento à lista bidirecional antes do nó especificado.

Versão para adição de um elemento segundo o valor.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,        // nó antes do qual deve ser feita a adição
   T                    value        // elemento a ser adicionado
   );
```

Valor de retorno

Retorna o ponteiro para o nó adicionado.

Versão para adição de um elemento, como o do nó formado, segundo o valor.

```
bool AddBefore(
   CLinkedListNode<T>*  node,        // nó antes do qual deve ser feita a adição
   CLinkedListNode<T>*  new_node     // nó a ser adicionado
   );
```

Parâmetros

**node*

[in] Nó da lista bidirecional, antes do qual será adicionado o novo elemento.

value

[in] Elemento para adição.

**new_node*

[in] Nó para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AddFirst

Adiciona um elemento ao topo da lista bidirecional.

Versão para adição de um elemento segundo o valor.

```
CLinkedListNode<T>* AddFirst(
   T   value                    // elemento para adição
   );
```

Valor de retorno

Retorna o ponteiro para o nó adicionado.

Versão para adição de um elemento, como o do nó formado, segundo o valor.

```
bool AddFirst(
   CLinkedListNode<T>*  node     // nó para adição
   );
```

Parâmetros

value

[in] Elemento para adição.

**node*

[in] Nó para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AddLast

Adiciona um elemento ao final da lista bidirecional.

Versão para adição de um elemento segundo o valor.

```
CLinkedListNode<T>* AddLast(
   T  value                      // elemento para adição
   );
```

Valor de retorno

Retorna o ponteiro para o nó adicionado.

Versão para adição de um elemento, como o do nó formado, segundo o valor.

```
bool AddLast(
   CLinkedListNode<T>*  node     // nó para adição
   );
```

Parâmetros

value

[in] Elemento para adição.

**node*

[in] Nó para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na lista bidirecional.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Head

Retorna o ponteiro para o primeiro nó na lista bidirecional.

```
CLinkedListNode<T>* Head();
```

Valor de retorno

Retorna o ponteiro para o primeiro nó.

© 2000-2024, MetaQuotes Ltd.

First

Retorna o ponteiro para o primeiro nó na lista bidirecional.

```
CLinkedListNode<T>* First();
```

Valor de retorno

Retorna o ponteiro para o primeiro nó.

© 2000-2024, MetaQuotes Ltd.

Last

Retorna o ponteiro para o último nó na lista bidirecional.

```
CLinkedListNode<T>* Last();
```

Valor de retorno

Retorna o ponteiro para o último nó.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a lista bidirecional contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a lista bidirecional contém um elemento com o valor especificado, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da lista bidirecional para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados os elementos da lista bidirecional.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da coleção.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do elemento especificado da lista bidirecional.

Versão para remoção de um elemento segundo o valor.

```
bool Remove(
   T  item                        // valor do elemento
   );
```

Versão para remover o elemento segundo o ponteiro para o nó.

```
bool Remove(
   CLinkedListNode<T>*  node      // nó do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

**node*

[in] Nó do elemento que é necessário remover.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveFirst

Remove o primeiro elemento da lista bidirecional.

```
bool RemoveFirst();
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveLast

Remove o último elemento da lista bidirecional.

```
bool RemoveLast();
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Find

Procura a primeira ocorrência de um valor numa matriz bidirecional.

```
CLinkedListNode<T>* Find(
   T  value     // valor procurado
   );
```

Parâmetros

value

[in] Valor procurado.

Valor de retorno

Se for bem-sucedido, retorna o ponteiro para o primeiro nó encontrado que contém o valor desejado, caso contrário, NULL.

© 2000-2024, MetaQuotes Ltd.

FindLast

Procura a última ocorrência de um valor numa matriz bidirecional.

```
CLinkedListNode<T>* FindLast(
   T  value     // valor procurado
   );
```

Parâmetros

value

[in] Valor procurado.

Valor de retorno

Se for bem-sucedido, retorna o ponteiro para o último nó encontrado que contém o valor desejado, caso contrário, NULL.

© 2000-2024, MetaQuotes Ltd.

CQueue<T>

Classe CQueue<T> é uma classe genérica que implementa a interface ICollection<T>.

Descrição

Classe CQueue<T> é uma coleção dinâmica de dados de tipo T, organizada como uma lista, que funciona com o princípio "primeira entrada-primeira saída" (FIFO).

Declaração

```
template<typename T>
class CQueue : public ICollection<T>
```

Cabeçalho

```
#include <Generic\Queue.mqh>
```

Hierarquia de herança

ICollection
  CQueue

Métodos de classe

Método	Descrição
Add	Adiciona um elemento à fila
Enqueue	Adiciona um elemento à fila
Count	Retorna o número de elementos na fila
Contains	Determina se a fila contém um elemento com o valor especificado
TrimExcess	Reduz a capacidade da fila para o número real de elementos, assim, liberando a memória não utilizada
CopyTo	Copia todos os elementos da fila para a matriz especificada, a partir do índice definido
Clear	Remove todos os elementos da fila
Remove	Remove a primeira ocorrência do elemento especificado da fila
Dequeue	Retorna o elemento inicial, removendo-a da fila
Peek	Retorna o elemento inicial sem removê-la da fila

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à fila.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Enqueue

Adiciona um elemento à fila.

```
bool Enqueue(
   T  value     // elemento para adição
   );
```

Parâmetros

value

[in] Elemento para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na fila.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a fila contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a fila contém um elemento com o valor especificado, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Reduz a capacidade da fila para o número real de elementos, assim, liberando a memória não utilizada.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da fila para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados os elementos da fila.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Ele remove todos os elementos da fila.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do elemento especificado da fila.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Dequeue

Retorna o elemento inicial, removendo-a da fila.

```
T Dequeue();
```

Valor de retorno

Retorna o elemento inicial.

© 2000-2024, MetaQuotes Ltd.

Peek

Retorna o elemento inicial sem removê-la da fila.

```
T Peek();
```

Valor de retorno

Retorna o elemento inicial.

© 2000-2024, MetaQuotes Ltd.

CRedBlackTree<T>

Classe CRedBlackTree<T> é uma classe genérica que implementa a interface ICollection<T>.

Descrição

Classe CRedBlackTree<T> é uma implementação da árvore vermelho-preto dinâmica, em cujos nós são armazenados dados do tipo T. A classe fornece os métodos principais para trabalhar com árvores vermelho-preto, isto é, adição, exclusão, pesquisa do valor máximo e mínimo, e outros.

Declaração

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Cabeçalho

```
#include <Generic\RedBlackTree.mqh>
```

Hierarquia de herança

ICollection
  CRedBlackTree

Métodos de classe

Método	Descrição
Add	Adiciona um elemento à árvore vermelho-preto
Root	Retorna o ponteiro para a raiz da árvore vermelho-preto
Count	Retorna o número de elementos na árvore vermelho-preto
Contains	Determina se a árvore vermelho-preto contém o elemento com o valor especificado
Comparer	Retorna o ponteiro para a interface IComparer<T> que é utilizada para a organização da árvore vermelho-preto.
TryGetMin	Recebe do elemento mínimo desde a árvore vermelho-preto
TryGetMax	Recebe o elemento máximo desde a árvore vermelho-preto
CopyTo	Copia todos os elementos da árvore vermelho-preto para a matriz especificada, a partir do índice definido
Clear	Remova todos os elementos da árvore vermelho-preto
Remove	Remove a ocorrência do elemento especificado da árvore vermelho-preto
RemoveMin	Remove o elemento com o valor mínimo da árvore vermelho-preto
RemoveMax	Remove o elemento com o valor máximo da árvore vermelho-preto

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
Find	Procura a ocorrência do valore especificado na árvore vermelho-preto
FindMax	Procura o elemento com o valor máximo na árvore vermelho-preto
FindMin	Procura o elemento com o valor mínimo na árvore vermelho-preto

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à árvore vermelho-preto.

```
bool Add(
   T  value     // elemento para adição
   );
```

Parâmetros

value

[in] Elemento para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na árvore vermelho-preto.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Root

Retorna o ponteiro para a raiz da árvore vermelho-preto.

```
CRedBlackTreeNode<T>* Root();
```

Valor de retorno

Retorna o ponteiro para a raiz.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a árvore vermelho-preto contém o elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a árvore vermelho-preto contém um elemento com o valor especificado, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retorna o ponteiro para a interface IComparer<T> que é utilizada para a organização da árvore vermelho-preto.

```
IComparer<T>* Comparer() const;
```

Valor de retorno

Retorna o ponteiro para a interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TryGetMin

Recebe do elemento mínimo desde a árvore vermelho-preto.

```
bool TryGetMin(
   T&  min     // variável para gravação do valor
   );
```

Parâmetros

&min

[out] Variável na qual será gravado o valor mínimo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TryGetMax

Recebe o elemento máximo desde a árvore vermelho-preto.

```
bool TryGetMax(
   T&  max     // variável para gravação do valor
   );
```

Parâmetros

&max

[out] Variável na qual será gravado o valor máximo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da árvore vermelho-preto para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados os elementos da árvore vermelho-preto.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remova todos os elementos da árvore vermelho-preto.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a ocorrência do elemento especificado da árvore vermelho-preto.

Versão para remoção de um elemento segundo o valor especificado.

```
bool Remove(
   T  value                        // valor do elemento
   );
```

Versão para remover o elemento segundo o ponteiro para o nó.

```
bool Remove(
   CRedBlackTreeNode<T>*  node     // nó do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

**node*

[in] Nó do elemento que é necessário remover.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveMin

Remove o elemento com o valor mínimo da árvore vermelho-preto.

```
bool RemoveMin();
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

RemoveMax

Remove o elemento com o valor máximo da árvore vermelho-preto.

```
bool RemoveMax();
```

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Find

Procura a ocorrência do valore especificado na árvore vermelho-preto.

```
CRedBlackTreeNode<T>* Find(
   T  value     // valor procurado
   );
```

Parâmetros

value

[in] Valor procurado.

Valor de retorno

Se for bem-sucedido, retorna o ponteiro para nó encontrado que contém o valor desejado, caso contrário, NULL.

© 2000-2024, MetaQuotes Ltd.

FindMin

Procura o elemento com o valor mínimo na árvore vermelho-preto.

```
CRedBlackTreeNode<T>* FindMin();
```

Valor de retorno

Se for bem-sucedido, retorna o ponteiro que contém o valor mínimo, caso contrário, NULL.

© 2000-2024, MetaQuotes Ltd.

FindMax

Procura o elemento com o valor máximo na árvore vermelho-preto.

```
CRedBlackTreeNode<T>* FindMax();
```

Valor de retorno

Se for bem-sucedido, retorna o ponteiro que contém o valor máximo, caso contrário, NULL.

© 2000-2024, MetaQuotes Ltd.

CSortedMap<TKey,TValue>

Classe CSortedMap<TKey, TValue> é uma classe genérica que implementa a interface IMap<TKey, TValue>.

Descrição

Classe CSortedMap<TKey, TValue> é uma implementação da tabela de hash dinâmica, cujos dados são armazenados como pares chave-valor, ordenados por chave e em conformidade com os requisitos de exclusividade de chave. Essa classe fornece os métodos básicos para trabalhar com uma tabela de hash, isto é, acesso ao valor da chave, procurar e eliminar par chave-valor e outros.

Declaração

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Cabeçalho

```
#include <Generic\SortedMap.mqh>
```

Hierarquia de herança

ICollection
  IMap
    CSortedMap

Métodos de classe

Método	Descrição
Add	Adiciona o par chave-valor à tabela de hash
Count	Retorna o número de elementos na tabela de hash classificada
Contains	Determina se a lista de hash classificada contém o par chave-valor especificado.
ContainsKey	Determina se a lista de hash classificada contém o par chave-valor com a chave especificada
ContainsValue	Determina se a lista de hash classificada contém o par chave-valor com o valor especificado
Comparer	Retorna o ponteiro para a interface IComparer<T> que é utilizada para a classificação da tabela de hash
CopyTo	Copia todos os pares chave-valor da tabela de hash classificada nas matrizes especificadas, a partir do índice especificado
Clear	Remove todos os elementos da tabela de hash classificada
Remove	Remove a primeira ocorrência do par chave-valor da tabela de hash classificada

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
TryGetValue	Obtém o elemento da tabela de hash classificada para a chave especificada
TrySetValue	Altera o valor do par chave-valor da tabela de hash classificada para a chave especificada

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona o par chave-valor à tabela de hash.

Versão para adicionar o par chave-valor gerado.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // par chave-valor
   );
```

Versão para adicionar o novo par chave-valor com a chave e valor especificado.

```
bool Add(
   TKey    key,                         // chave
   TValue  value                        // valor
   );
```

Parâmetros

pair

[in] Par chave-valor.

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na tabela de hash classificada.

```
int Count();
```

© 2000-2024, MetaQuotes Ltd.

Comparer

Retorna o ponteiro para a interface IComparer<T> que é utilizada para a classificação da tabela de hash.

```
IComparer<TKey>* Comparer() const;
```

Valor de retorno

Retorna o ponteiro para a interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a lista de hash classificada contém o par chave-valor especificado.

Versão para trabalhar com o par chave-valor gerado.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // par chave-valor
   );
```

Versão para trabalhar com o par chave-valor sob a forma de uma chave e valor especificados separadamente.

```
bool Contains(
   TKey    key,                         // chave
   TValue  value                        // valor
   );
```

Parâmetros

*item

[in] Par chave-valor.

key

[in] Chave.

value

[in] Valor.

Valor de retorno

Retorna true, se na tabela de hash classificada houver um par chave-valor com a chave e valor especificados, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ContainsKey

Determina se a lista de hash classificada contém o par chave-valor com a chave especificada.

```
bool ContainsKey(
   TKey  key     // chave
   );
```

Parâmetros

key

[in] Chave.

Valor de retorno

Retorna true, se na tabela de hash classificada houver um par chave-valor com a chave especificada, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ContainsValue

Determina se a lista de hash classificada contém o par chave-valor com o valor especificado.

```
bool ContainsValue(
   TValue  value     // valor
   );
```

Parâmetros

value

[in] Valor.

Valor de retorno

Retorna true, se na tabela de hash classificada houver um par chave-valor com o valor especificado, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os pares chave-valor da tabela de hash classificada nas matrizes especificadas, a partir do índice especificado.

Versão para cópia da tabela de hash na matriz dos pares chave-valor.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // matriz para gravação dos pares cha
   const int                    dst_start=0      // índice de partida para gravação
   );
```

Versão para cópia da tabela de hash classificada em matrizes separadas para as chaves e valores.

```
int CopyTo(
   TKey&      dst_keys[],                        // matriz para gravação das chaves
   TValue&    dst_values[],                      // matriz para gravação de valores
   const int  dst_start=0                        // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que serão gravados todos os pares da tabela de hash.

&dst_keys[]

[out] Matriz em que serão gravadas todas as chaves da tabela de hash.

&dst_values[]

[out] Matriz em que serão armazenados todos os valores da tabela de hash.

dst_start=0

[in] Índice nas matrizes, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de pares chave-valor copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da tabela de hash classificada.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do par chave-valor da tabela de hash classificada.

Versão para remover o par chave-valor após formado o par chave-valor.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // par chave-valor
   );
```

Versão para remover o par chave-valor segundo a chave.

```
bool Remove(
   TKey  key                            // chave
   );
```

Parâmetros

item

[in] Par chave-valor.

key

[in] Chave.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TryGetValue

Obtém o elemento da tabela de hash classificada para a chave especificada.

```
bool TryGetValue(
   TKey     key,       // chave
   TValue&  value      // variável para gravação do valor
   );
```

Parâmetros

key

[in] Chave.

&value

[out] Variável em que será gravado o valor especificado do par chave-valor.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TrySetValue

Altera o valor do par chave-valor da tabela de hash classificada para a chave especificada.

```
bool TrySetValue(
   TKey    key,       // chave
   TValue  value      // novo valor
   );
```

Parâmetros

key

[in] Chave.

value

[in] Novo valor que deve ser atribuído ao par chave-valor especificado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CSortedSet<T>

Classe CSortedSet<T> é uma classe genérica que implementa a interface ISet<T>.

Descrição

Classe CSortedSet<T> é uma implementação do conjunto de dados dinâmico de tipo T ordenado, em conformidade com os requisitos de exclusividade de cada valor. Essa classe fornece os métodos básicos para trabalhar com conjuntos e operações sobre eles, isto é, união de conjuntos e interseção de conjuntos, definição de subconjuntos estritos e não estritos e outros.

Declaração

```
template<typename T>
class CSortedSet : public ISet<T>
```

Cabeçalho

```
#include <Generic\SortedSet.mqh>
```

Hierarquia de herança

ICollection
  ISet
    CSortedSet

Métodos de classe

Método	Descrição
Add	Adiciona um elemento ao conjunto classificado
Count	Retorna o número de elementos no conjunto classificado
Contains	Determina se o conjunto classificado contém um elemento com o valor especificado
Comparer	Retorna o ponteiro para a interface IComparer<T> que é utilizada para a classificação do conjunto classificado
TryGetMin	Obtém o elemento mínimo do conjunto classificado
TryGetMax	Recebe o elemento máximo do conjunto classificado
CopyTo	Copia todos os elementos do conjunto classificado para a matriz especificada, a partir do índice especificado
Clear	Remove todos elementos do conjunto classificado
Remove	Remove a ocorrência do elemento especificado do conjunto classificado
ExceptWith	Executa a operação de diferença da coleção atual e coleção transmitida (matriz)

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
IntersectWith	Executa a operação de interseção da coleção atual e coleção transmitida (matriz)
SymmetricExceptWith	Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz)
UnionWith	Executa a operação de união da coleção atual e coleção transmitida (matriz)
IsProperSubsetOf	Determina se o conjunto atual classificado é subconjunto estrito da coleção especificada ou matriz
IsProperSupersetOf	Determina se o conjunto atual classificado é superconjunto estrito da coleção especificada ou matriz
IsSubsetOf	Determina se o conjunto atual classificado é subconjunto da coleção especificada ou matriz
IsSupersetOf	Determina se o conjunto atual classificado é superconjunto da coleção especificada ou matriz
Overlaps	Determina se o conjunto atual classificado se cruza com a coleção especificada ou matriz
SetEquals	Determina se o conjunto atual classificado contém todos os elementos da coleção especificada ou matriz
GetViewBetween	Recebe, a partir do conjunto atual classificado, o subconjunto definido pelos valores mínimo e máximo
GetReverse	Recebe uma cópia do conjunto atual classificado em que todos os elementos estão dispostos na ordem inversa

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento ao conjunto classificado.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos no conjunto classificado.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se o conjunto classificado contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se o conjunto contém um elemento com o valor especificado, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

Comparer

Retorna o ponteiro para a interface IComparer<T> que é utilizada para a classificação do conjunto classificado.

```
IComparer<T>* Comparer() const;
```

Valor de retorno

Retorna o ponteiro para a interface IComparer<T>.

© 2000-2024, MetaQuotes Ltd.

TryGetMin

Obtém o elemento mínimo do conjunto classificado.

```
bool TryGetMin(
   T&  min     // variável para gravação do valor
   );
```

Parâmetros

&min

[out] Variável na qual será gravado o valor mínimo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

TryGetMax

Recebe o elemento máximo do conjunto classificado.

```
bool TryGetMax(
   T&  max     // variável para gravação do valor
   );
```

Parâmetros

&max

[out] Variável na qual será gravado o valor máximo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos do conjunto classificado para a matriz especificada, a partir do índice especificado.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que são gravados os elementos do conjunto.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos elementos do conjunto classificado.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a ocorrência do elemento especificado do conjunto classificado.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ExceptWith

Executa a operação de diferença da coleção atual e coleção transmitida (matriz). Ou seja, remove da coleção atual (matriz) todos os elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void ExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void ExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção que será subtraída a partir do conjunto atual classificado.

&collection[]

[in] Matriz que será subtraída a partir do conjunto atual classificado.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IntersectWith

Executa a operação de interseção da coleção atual e coleção transmitida (matriz). Ou seja, deixa na coleção atual apenas aqueles elementos que estão também na coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void IntersectWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void IntersectWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a multiplicação.

&collection[]

[in] Matriz a partir da qual será construída a multiplicação.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

SymmetricExceptWith

Executa a operação de diferença simétrica da coleção atual e coleção transmitida (matriz). Ou seja, na coleção atual conterá apenas aqueles elementos que estavam no objeto original ou no transmitido, mas não ao mesmo tempo em ambos.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void SymmetricExceptWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual é construída a diferença simétrica.

&collection[]

[in] Matriz a partir da qual é construída a diferença simétrica.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

UnionWith

Executa a operação de união da coleção atual e coleção transmitida (matriz). Ou seja, adiciona à coleção atual (matriz) os elementos em falta da coleção especificada (matriz).

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
void UnionWith(
   ICollection<T>*  collection     // coleção
   );
```

Versão para trabalhar com a matriz.

```
void UnionWith(
   T&  array[]                     // matriz
   );
```

Parâmetros

**collection*

[in] Coleção a partir da qual será construída a soma.

&collection[]

[in] Matriz a partir da qual será construída a soma.

Observação

O resultado é gravado na coleção atual (matriz).

© 2000-2024, MetaQuotes Ltd.

IsProperSubsetOf

Determina se o conjunto atual classificado é subconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual classificado for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsProperSupersetOf

Determina se o conjunto atual classificado é superconjunto estrito da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsProperSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual classificado for um superconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSubsetOf

Determina se o conjunto atual classificado é subconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSubsetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual classificado for um subconjunto estrito, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsSupersetOf

Determina se o conjunto atual classificado é superconjunto da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // coleção para determinar a relação
   );
```

Versão para trabalhar com a matriz.

```
bool IsSupersetOf(
   T&  array[]                     // matriz para determinar a relação
   );
```

Parâmetros

**collection*

[in] Coleção para definir a relação.

&collection[]

[in] Matriz para definir a relação.

Valor de retorno

Retorna true, se o conjunto atual classificado for um superconjunto, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Overlaps

Determina se o conjunto atual classificado se cruza com a coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool Overlaps(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool Overlaps(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para definir o cruzamento.

&collection[]

[in] Matriz para definir o cruzamento.

Valor de retorno

Retorna true, se houver uma interseção entre o conjunto atual classificado e a coleção ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SetEquals

Determina se o conjunto atual classificado contém todos os elementos da coleção especificada ou matriz.

Versão para trabalhar com a coleção que implementa a interface ICollection<T>.

```
bool SetEquals(
   ICollection<T>*  collection     // coleção para comparação
   );
```

Versão para trabalhar com a matriz.

```
bool SetEquals(
   T&  array[]                     // matriz para comparação
   );
```

Parâmetros

**collection*

[in] Coleção para associação de elementos.

&collection[]

[in] Matriz para associação de elementos.

Valor de retorno

Retorna true, se, no conjunto atual classificado, estiverem todos os elementos da coleção especificada ou matriz, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

GetViewBetween

Recebe, a partir do conjunto atual classificado, o subconjunto definido pelos valores mínimo e máximo.

```
bool GetViewBetween(
   T&  array[],          // matriz para gravação
   T   lower_value,      // valor mínimo
   T   upper_value       // valor máximo
   );
```

Parâmetros

&array[]

[out] Matriz para armazenamento do subconjunto.

lower_value

[in] Valor mínimo do intervalo.

upper_value

[in] Valor máximo do intervalo.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

GetReverse

Recebe uma cópia do conjunto atual classificado em que todos os elementos estão dispostos na ordem inversa.

```
bool GetReverse(
   T&  array[]     // matriz para gravação
   );
```

Parâmetros

&array[]

[out] Matriz para gravação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

CStack<T>

Classe CStack<T> é uma classe genérica que implementa a interface ICollection<T>.

Descrição

Classe CStack<T> é uma coleção dinâmica de dados de tipo T, organizada como uma pilha, que funciona com o princípio "última entrada-primeira saída" (LIFO).

Declaração

```
template<typename T>
class CStack : public ICollection<T>
```

Cabeçalho

```
#include <Generic\Stack.mqh>
```

Hierarquia de herança

ICollection
  CStack

Métodos de classe

Método	Descrição
Add	Adiciona um elemento à pilha
Count	Retorna o número de elementos na pilha
Contains	Determina se a pilha contém um elemento com o valor especificado
TrimExcess	Reduz a capacidade de pilha para o número real de elementos
CopyTo	Copia todos os elementos da pilha para a matriz especificada, a partir do índice definido
Clear	Remove todos os elementos da pilha
Remove	Remove a primeira ocorrência do elemento especificado da pilha
Push	Adiciona um elemento à pilha
Peek	Retorna o elemento inicial sem removê-la da pilha
Pop	Retorna o elemento inicial sem removê-la da pilha

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um elemento à pilha.

```
bool Add(
   T  value     // valor do elemento
   );
```

Parâmetros

value

[in] Valor do elemento a ser adicionado.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Count

Retorna o número de elementos na pilha.

```
int Count();
```

Valor de retorno

Retorna o número de elementos.

© 2000-2024, MetaQuotes Ltd.

Contains

Determina se a pilha contém um elemento com o valor especificado.

```
bool Contains(
   T  item     // valor procurado
   );
```

Parâmetros

item

[in] Valor procurado.

Valor de retorno

Retorna true, se a pilha contém um elemento com o valor especificado, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

TrimExcess

Reduz a capacidade da pilha para o número real de elementos, assim, liberando a memória não utilizada.

```
void TrimExcess();
```

© 2000-2024, MetaQuotes Ltd.

CopyTo

Copia todos os elementos da pilha para a matriz especificada, a partir do índice definido.

```
int CopyTo(
   T&          dst_array[],      // matriz para gravação
   const int   dst_start=0       // índice de partida para gravação
   );
```

Parâmetros

&dst_array[]

[out] Matriz em que são gravados os elementos da pilha.

dst_start=0

[in] Índice na matriz, a partir do qual começa a cópia.

Valor de retorno

Retorna o número de elementos copiados.

© 2000-2024, MetaQuotes Ltd.

Clear

Remove todos os elementos da pilha.

```
void Clear();
```

© 2000-2024, MetaQuotes Ltd.

Remove

Remove a primeira ocorrência do elemento especificado da pilha.

```
bool Remove(
   T  item     // valor do elemento
   );
```

Parâmetros

item

[in] Valor do elemento que deve ser removido.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Push

Adiciona um elemento à pilha.

```
bool Push(
   T  value     // elemento para adição
   );
```

Parâmetros

value

[in] Elemento para adição.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Peek

Retorna o elemento inicial sem removê-la da pilha.

```
T Peek();
```

Valor de retorno

Retorna o elemento inicial.

© 2000-2024, MetaQuotes Ltd.

Pop

Retorna o elemento inicial sem removê-la da pilha.

```
T Pop();
```

Valor de retorno

Retorna o elemento inicial.

© 2000-2024, MetaQuotes Ltd.

ArrayBinarySearch

Procura o valor especificado numa matriz unidimensional classificada em ordem ascendente usando a interface IComparable<T> para comparação de elementos.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // matriz para pesquisa
   const int       start_index,      // índice de partida
   const int       count,            // intervalo de pesquisa
   T               value,            // valor procurado
   IComparer<T>*   comparer          // interface para comparação
   );
```

Parâmetros

&array[]

[out] Matriz para pesquisa.

value

[in] Valor procurado.

**comparer*

[in] Interface para comparação de elementos.

start_index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

Valor de retorno

Retorna o índice do elemento encontrado. Se o valor procurado não for encontrado, ele retornará o índice do elemento menor, cujo valor é mais próximo de todos.

© 2000-2024, MetaQuotes Ltd.

ArrayIndexOf

Procura a primeira ocorrência de um valor numa matriz unidimensional

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // matriz para pesquisa
   T           value,            // valor procurado
   const int   start_index,      // índice de partida
   const int   count             // intervalo de pesquisa
   );
```

Parâmetros

&array[]

[out] Matriz para pesquisa.

value

[in] Valor procurado.

start_index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

Valor de retorno

Retorna o índice do primeiro elemento encontrado.. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

ArrayLastIndexOf

Procura a última ocorrência de um valor numa matriz unidimensional.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // matriz para pesquisa
   T           value,            // valor procurado
   const int   start_index,      // índice de partida
   const int   count             // intervalo de pesquisa
   );
```

Parâmetros

&array[]

[out] Matriz para pesquisa.

value

[in] Valor procurado.

start_index

[in] Índice de partida, a partir do qual começa a pesquisa.

count

[in] Comprimento do intervalo de pesquisa.

Valor de retorno

Retorna o índice do último elemento encontrado. Se o valor não for encontrado, retornará -1.

© 2000-2024, MetaQuotes Ltd.

ArrayReverse

Altera o sequenciamento dos elementos numa matriz unidimensional.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // matriz de origem
   const int   start_index,      // índice de partida
   const int   count             // número de elementos
   );
```

Parâmetros

&array[]

[out] Matriz de origem.

start_index

[in] Índice inicial.

count

[in] Número de elementos da matriz envolvidos na operação.

Valor de retorno

Retorna true em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Compare

Compara dois valores no que diz respeito à relação "maior, menor ou igual".

Versão para comparar dois valores de tipo bool.

```
int Compare(
   const bool  x,          // primeiro valor
   const bool  y           // segundo valor
   );
```

Versão para comparar dois valores de tipo char.

```
int Compare(
   const char  x,          // primeiro valor
   const char  y           // segundo valor
   );
```

Versão para comparar dois valores de tipo uchar.

```
int Compare(
   const uchar  x,         // primeiro valor
   const uchar  y          // segundo valor
   );
```

Versão para comparar dois valores de tipo short.

```
int Compare(
   const short  x,         // primeiro valor
   const short  y          // segundo valor
   );
```

Versão para comparar dois valores de tipo ushort.

```
int Compare(
   const ushort  x,        // primeiro valor
   const ushort  y         // segundo valor
   );
```

Versão para comparar dois valores de tipo color.

```
int Compare(
   const color  x,         // primeiro valor
   const color  y          // segundo valor
   );
```

Versão para comparar dois valores de tipo int.

```
int Compare(
   const int  x,           // primeiro valor
   const int  y            // segundo valor
   );
```

© 2000-2024, MetaQuotes Ltd.

Versão para comparar dois valores de tipo uint.

```
int Compare(
   const uint  x,          // primeiro valor
   const uint  y           // segundo valor
   );
```

Versão para comparar dois valores de tipo datetime.

```
int Compare(
   const datetime  x,      // primeiro valor
   const datetime  y       // segundo valor
   );
```

Versão para comparar dois valores de tipo long.

```
int Compare(
   const long  x,          // primeiro valor
   const long  y           // segundo valor
   );
```

Versão para comparar dois valores de tipo ulong.

```
int Compare(
   const ulong  x,         // primeiro valor
   const ulong  y          // segundo valor
   );
```

Versão para comparar dois valores de tipo float.

```
int Compare(
   const float  x,         // primeiro valor
   const float  y          // segundo valor
   );
```

Versão para comparar dois valores de tipo double.

```
int Compare(
   const double  x,        // primeiro valor
   const double  y         // segundo valor
   );
```

Versão para comparar dois valores de tipo string.

```
int Compare(
   const string  x,        // primeiro valor
   const string  y         // segundo valor
   );
```

Versão para comparar dois valores de outros tipos.

```
template<typename T>
int Compare(
```

© 2000-2024, MetaQuotes Ltd.

```
    T   x,                  // primeiro valor
    T   y                   // segundo valor
    );
```

Parâmetros

x

[in] Primeiro valor.

y

[in] Segundo valor

Valor de retorno

Retorna o número que expressa a razão entre os dois valores comparados:

- resultado inferior a zero — x é menor que y (x<y)
- resultado igual a zero — x igual a y (x=y)
- resultado superior a zero — x maior que y (x>y)

Observação

Se o tipo T é um objeto que implementa a interface IComparable<T>, os objetos são comparados com base em seu método de comparação Compare. Em todos os outros casos, retorna-se 0.

© 2000-2024, MetaQuotes Ltd.

Equals

Compara se dois valores são iguais.

```
template<typename T>
bool Equals(
   T  x,      // primeiro valor
   T  y       // segundo valor
   );
```

Parâmetros

x

[in] Primeiro valor.

y

[in] Segundo valor.

Valor de retorno

Retorna true, se os objetos são iguais, caso contrário, false.

Observação

Se o tipo T é um objeto que implementa a interface IEqualityComparable<T>, os objetos são comparados com base em seu método de comparação Equals. Em todos os outros casos, é implementada a comparação padrão para a igualdade.

© 2000-2024, MetaQuotes Ltd.

GetHashCode

Calcula o valor de código hash.

Versão para trabalhar com o tipo bool.

```
int GetHashCode(
   const bool  value          // valor
   );
```

Versão para trabalhar com o tipo char.

```
int GetHashCode(
   const char  value          // valor
   );
```

Versão para trabalhar com o tipo uchar.

```
int GetHashCode(
   const uchar  value         // valor
   );
```

Versão para trabalhar com o tipo short.

```
int GetHashCode(
   const short  value         // valor
   );
```

Versão para trabalhar com o tipo ushort.

```
int GetHashCode(
   const ushort  value        // valor
   );
```

Versão para trabalhar com o tipo color.

```
int GetHashCode(
   const color  value         // valor
   );
```

Versão para trabalhar com o tipo int.

```
int GetHashCode(
   const int  value           // valor
   );
```

Versão para trabalhar com o tipo uint.

```
int GetHashCode(
   const uint  value          // valor
   );
```

Versão para trabalhar com o tipo datetime.

```
int GetHashCode(
```

© 2000-2024, MetaQuotes Ltd.

```
   const datetime  value     // valor
   );
```

Versão para trabalhar com o tipo long.

```
int GetHashCode(
   const long  value         // valor
   );
```

Versão para trabalhar com o tipo ulong.

```
int GetHashCode(
   const ulong  value        // valor
   );
```

Versão para trabalhar com o tipo float.

```
int GetHashCode(
   const float  value        // valor
   );
```

Versão para trabalhar com o tipo double.

```
int GetHashCode(
   const double  value       // valor
   );
```

Versão para trabalhar com o tipo string.

```
int GetHashCode(
   const string  value       // valor
   );
```

Versão para trabalhar com os outros tipos.

```
template<typename T>
int GetHashCode(
   T  value                  // valor
   );
```

Parâmetros

value

[in] O valor para o qual é necessário obter o código hash.

Valor de retorno

Retorna o código hash.

Observação

Se o tipo T é um objeto que implementa a interface IEqualityComparable<T>, o código hash é obtido com base em seu método HashCode. Em todos os outros casos, o código hash é calculado como o valor do código hash a partir do nome do tipo desse valor.

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Operações de Arquivo

Esta seção contém os detalhes técnicos das classes para operações de arquivo e descrições dos componentes correspondentes na biblioteca padrão MQL5.

As classes para operações de arquivo pouparão tempo no desenvolvimento de aplicações que utilizam operações de entrada/saída de arquivo.

A Biblioteca Padrão MQL5 é colocada no diretório de trabalho do terminal na pasta Include\Files.

Classe	Descrição
CFile	Base de dados da classe para operações de arquivo
CFileBin	Classe para operações de arquivo binário
CFileTxt	Texto na classe para operações de arquivo

© 2000-2024, MetaQuotes Ltd.

CFile

CFile é uma classe base para CFileBin e CFileTxt.

Descrição

Classe CFile proporciona o acesso simplificado a todos os seus descendentes para funções de arquivo e pasta API MQL5.

Declaração

```
class CFile: public CObject
```

Título

```
#include <Files\File.mqh>
```

Hierarquia de herança

CObject
  CFile

Descendentes diretos

CFileBin, CFilePipe, CFileTxt

Métodos de classe

Atributos	
Handle	Obtém manipulador de arquivo
Filename	Obtém um nome de arquivo
Flags	Obtém os flags de arquivo
SetUnicode	Define/Limpa o flag FILE_UNICODE
SetCommon	Define/Limpa o flag FILE_COMMON
Os métodos gerais para arquivos	
Open	Abre arquivo
Close	Fecha arquivo
Delete	Exclui arquivo
IsExist	Verifica existência de arquivo
Copy	Copia arquivos
Move	Renomeia/move arquivo
Size	Obtém o tamanho do arquivo

© 2000-2024, MetaQuotes Ltd.

Atributos	
Tell	Obtém posição atual do arquivo
Seek	Define posição atual do arquivo
Flush	Liberação de dados no disco
IsEnding	Verifica arquivo para final
IsLineEnding	Os testes de linha para finalizar
Os métodos gerais para pastas	
FolderCreate	Cria pasta
FolderDelete	Exclui pasta
FolderClean	Limpa pasta
Os métodos de pesquisa	
FileFindFirst	Inicia a pesquisa de arquivo
FileFindNext	Continua a pesquisa de arquivo
FileFindClose	Fecha a pesquisa manipulada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

As classes derivadas:

- CFileBin
- CFileTxt

© 2000-2024, MetaQuotes Ltd.

Handle

Obtém manipulador do arquivo aberto.

```
int  Handle()
```

Valor de retorno

Manipulador do arquivo aberto, atribuído à instância de classe. Se não houver nenhum arquivo atribuído, ele retorna -1.

© 2000-2024, MetaQuotes Ltd.

FileName

Obtém o nome do arquivo arquivo aberto.

```
string  FileName()
```

Valor de retorno

O nome do arquivo aberto, atribuído à instância de classe. Se não houver nenhum arquivo atribuído, retorna "" (duas aspas).

© 2000-2024, MetaQuotes Ltd.

Flags

Obtém flags do arquivo aberto.

```
int  Flags()
```

Valor de retorno

Flags do arquivo aberto, atribuídos à instância de classe.

© 2000-2024, MetaQuotes Ltd.

SetUnicode

Define/limpa o flag FILE_UNICODE.

```
void  SetUnicode(
   bool  unicode      // New flag value
   )
```

Parâmetros

unicode

[in] O novo valor para o flag FILE_UNICODE.

Observação

O resultado das operações string depende do flag FILE_UNICODE. Se for falso, os códigos ANSI são utilizados (símbolos de um byte). Se for definido, os códigos UNICODE são usados (símbolos de dois bytes). Se o arquivo foi aberto, o flag não pode ser alterado.

© 2000-2024, MetaQuotes Ltd.

SetCommon

Define/Limpa o flag FILE_COMMON.

```
void  SetCommon(
   bool  common      // New flag value
   )
```

Parâmetros

common

[in] O novo valor para o flag FILE_COMMON.

Observação

O flag FILE_UNICODE determina a pasta de trabalho atual. Se for falso, a pasta do terminal local é usada como a pasta atual de trabalho. Se for verdadeiro, a pasta geral é usada como a pasta atual de trabalho. Se o arquivo foi aberto, o flag não pode ser alterado.

© 2000-2024, MetaQuotes Ltd.

Open

Se com sucesso, abre o arquivo determinado e o atribui para a instância da classe.

```
int  Open(
   const string  file_name,        // File name
   int           flags,            // Flags
   short         delimiter=9       // Separator
   )
```

Parâmetros

file_name

[in] O nome do arquivo para abrir.

flags

[in] Flags de arquivo aberto

delimiter=9

[in] Separador do arquivo CSV.

Valor de retorno

Manipulador do arquivo aberto.

Observação

A pasta de trabalho depende do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

Close

Fecha arquivo, atribuído à instância de classe.

```
void  Close()
```

© 2000-2024, MetaQuotes Ltd.

Delete

Exclui o arquivo, atribuído à instância de arquivo.

```
void  Delete()
```

Delete

Exclui o arquivo especificado.

```
void  Delete(
   const string  file_name      // File name
   )
```

Parâmetros

file_name

[in] O nome do arquivo do arquivo a ser excluído.

Observação

A pasta de trabalho é dependente do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

IsExist

Verifica existência de arquivo

```
bool  IsExist(
   const string  file_name      // File name
   )
```

Parâmetros

file_name

[in] Nome do arquivo a verificar.

Valor de retorno

verdadeiro se o arquivo existe.

© 2000-2024, MetaQuotes Ltd.

Copy

Copia um arquivo.

```
bool  Copy(
   const string  src_name,       // Source file name
   int           src_flag,       // Flag
   const string  dst_name,       // Destination file name
   int           dst_flags       // Flags
   )
```

Parâmetros

src_name

[in] O nome do arquivo para copiar.

src_flag

[in] Flags do arquivo para copiar (apenas FILE_COMMON é usado).

dst_name

[in] Nome do arquivo de destino.

dst_flags

[in] Flags do arquivo de destino (apenas FILE_REWRITE e FILE_COMMON são utilizados).

Valor de retorno

verdadeiro se com sucesso, falso se não tiver sido copiado.

© 2000-2024, MetaQuotes Ltd.

Move

Renomeia/move arquivo.

```
bool  Move(
   const string  src_name,       // Source file name
   int           src_flag,       // Flag
   const string  dst_name,       // Destination file name
   int           dst_flags       // Flags
   )
```

Parâmetros

src_name

[in] O nome do arquivo para mover.

src_flag

[in] Flags do arquivo para copiar (apenas FILE_COMMON é usado).

dst_name

[in] Nome do arquivo de destino.

dst_flags

[in] Flags do arquivo de destino (apenas FILE_REWRITE e FILE_COMMON são utilizados).

Valor de retorno

verdadeiro se teve sucesso. Falso se não foi movido.

© 2000-2024, MetaQuotes Ltd.

Size

Obtém o tamanho do arquivo em bytes.

```
ulong  Size()
```

Valor de retorno

Tamanho do arquivo em bytes. Se não houver qualquer arquivo atribuído, retorna ULONG_MAX.

© 2000-2024, MetaQuotes Ltd.

Tell

Obtém a posição atual do arquivo.

```
ulong  Tell()
```

Valor de retorno

A posição atual do arquivo. Se não houver qualquer arquivo atribuído, retorna ULONG_MAX.

© 2000-2024, MetaQuotes Ltd.

Seek

Define posição atual do arquivo.

```
void  Seek(
   long                offset,      // Offset
   ENUM_FILE_POSITION  origin       // Origin
   )
```

Parâmetros

offset

[in] Deslocamento em bytes do arquivo (pode ser negativo).

origin

[in] Origem do deslocamento.

Valor de retorno

verdadeiro se teve sucesso. Falso se a posição do arquivo não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Flush

Libera todos os dados do buffer de entrada/saída de arquivo no disco.

```
void  Flush()
```

© 2000-2024, MetaQuotes Ltd.

IsEnding

Verifica arquivo para finalizar. É usado durante as operações de leitura do arquivo.

```
bool  IsEnding()
```

Valor de retorno

verdadeiro se foi finalizado o arquivo depois de ler ou da operação de busca.

© 2000-2024, MetaQuotes Ltd.

IsLineEnding

Verifica o arquivo para o final da linha. É usado durante as operações de leitura do arquivo.

```
bool  IsLineEnding()
```

Valor de retorno

verdadeiro se o final da linha foi alcançado após a operação de leitura do arquivo txt ou csv (caracteres CR-LF).

© 2000-2024, MetaQuotes Ltd.

FolderCreate

Cria nova pasta.

```
bool  FolderCreate(
   const string  folder_name      // Folder name
   )
```

Parâmetros

folder_name

[in] Nome da pasta para criar. Contém caminho para a pasta em relação à pasta definida pelo flag FILE_COMMON.

Valor de retorno

verdadeiro se teve sucesso. Falso se a pasta não foi criada.

Observação

A pasta de trabalho é dependente do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

FolderDelete

Exclui pasta determinada.

```
bool  FolderDelete(
   const string  folder_name      // Folder name
   )
```

Parâmetros

folder_name

[in] Nome da pasta a ser excluída. Contém caminho para a pasta em relação à pasta definida pelo flag FILE_COMMON.

Valor de retorno

verdadeiro se teve sucesso. Falso se a pasta não foi excluída.

Observação

A pasta de trabalho é dependente do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

FolderClean

Limpa pasta determinada.

```
bool  FolderClean(
   const string  folder_name      // Folder name
   )
```

Parâmetros

folder_name

[in] Nome da pasta a ser excluída. Contém caminho para a pasta em relação à pasta definida pelo flag FILE_COMMON.

Valor de retorno

verdadeiro se teve sucesso. Falso se a pasta não foi limpa.

Observação

A pasta de trabalho é dependente do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

FileFindFirst

Inicia a busca de arquivos usando o filtro determinado.

```
int  FileFindFirst(
   const string  filter,         // Search Filter
   string&       file_name       // Reference to string
   )
```

Parâmetros

filter

[in] Filtro de pesquisa.

file_name

[out] A referência string para o primeiro arquivo encontrado.

Valor de retorno

Se for bem sucedido, ele retorna o manipulador, que poderá ser usado em nova pesquisa de arquivos usando o FileFindNext, ou INVALID_HANDLE se não houver qualquer arquivo correspondente ao filtro determinado.

Observação

A pasta de trabalho é dependente do flag FILE_COMMON, definido pelo método SetCommon().

© 2000-2024, MetaQuotes Ltd.

FileFindNext

Continua a pesquisa, iniciada pela função FileFindFirst().

```
bool  FileFindNext(
   int      search_handle,     // Search handle
   string&  file_name          // Reference to string for the next file found
   )
```

Parâmetros

search_handle

[in] Manipulador da pesquisa, retorno pelo método FileFindFirst().

file_name

[in] A referência string para o nome do arquivo encontrado em caso de êxito.

Valor de retorno

Verdadeiro se teve sucesso. Falso se não houver qualquer arquivo correspondente ao filtro determinado.

© 2000-2024, MetaQuotes Ltd.

FileFindClose

Fecha manipulador da pesquisa.

```
void  FileFindClose(
   int  search_handle      // Search handle
   )
```

Parâmetros

search_handle

[in] Manipulador da pesquisa, retorno pelo método FileFindFirst().

© 2000-2024, MetaQuotes Ltd.

CFileBin

CFileBin é uma classe de acesso simplificado para os arquivos binários

Descrição

Classe CFileBin fornece um acesso aos arquivos binários.

Declaração

```
class CFileBin: public CFile
```

Título

```
#include <Files\FileBin.mqh>
```

Hierarquia de herança

CObject
  CFile
    CFileBin

Métodos de classe

Abrir métodos	
Open	Abre um arquivo binário
Métodos de gravação	
WriteChar	Escreve variável do tipo char ou uchar
WriteShort	Escreve variável do tipo short ou ushort
WriteInteger	Escreve variável do tipo int ou uint
WriteLong	Escreve variável do tipo long ou ulong
WriteFloat	Escreve variável do tipo float
WriteDouble	Escreve variável do tipo double
WriteString	Escreve variável do tipo string
WriteCharArray	Escreve um array das variáveis do tipo char ou uchar
WriteShortArray	Escreve um array das variáveis do tipo short ou ushort
WriteIntegerArray	Escreve um array das variáveis do tipo int ou uint
WriteLongArray	Escreve um array das variáveis do tipo long ou ulong
WriteFloatArray	Escreve um array das variáveis do tipo float
WriteDoubleArray	Escreve um array das variáveis do tipo double
WriteObject	Escreve dados da instância de classe herdeira CObject

© 2000-2024, MetaQuotes Ltd.

Abrir métodos	
Lendo métodos	
ReadChar	Lê variável do tipo char ou uchar
ReadShort	Lê variável do tipo short ou ushort
ReadInteger	Lê variável do tipo int ou uint
ReadLong	Lê variável do tipo long ou ulong
ReadFloat	Lê variável do tipo float
ReadDouble	Lê variável do tipo double
ReadString	Lê variável do tipo string
ReadCharArray	Lê um array das variáveis do tipo char ou uchar
ReadShortArray	Lê um array das variáveis do tipo short ou ushort
ReadIntegerArray	Lê um array das variáveis do tipo int ou uint
ReadLongArray	Lê um array das variáveis do tipo long ou ulong
ReadFloatArray	Lê um array das variáveis do tipo tipo float
ReadDoubleArray	Lê um array das variáveis do tipo double
ReadObject	Lê dados da instância de classe herdeira CObject

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Open

Se com sucesso, abre o arquivo binário determinado e o atribui para a instância de classe.

```
int  Open(
   const string  file_name,      // File name
   int           flags           // Flags
   )
```

Parâmetros

file_name

[in] O nome do arquivo a ser aberto.

flags

[in] Flags de arquivo aberto (existe uma configuração forçada do flag FILE_BIN)

Valor de retorno

Manipulador do arquivo aberto.

© 2000-2024, MetaQuotes Ltd.

WriteChar

Escreve uma variável para arquivo do tipo char ou uchar

```
uint  WriteChar(
   char  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteShort

Escreve uma variável para arquivo do tipo short ou ushort.

```
uint  WriteShort(
   short  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteInteger

Escreve uma variável para arquivo do tipo int ou uint.

```
uint  WriteInteger(
   int  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteLong

Escreve uma variável para arquivo do tipo long ou ulong.

```
uint  WriteLong(
   long  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteFloat

Escreve uma variável para arquivo do tipo float.

```
uint  WriteFloat(
   float  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteDouble

Escreve uma variável para arquivo do tipo double .

```
uint  WriteDouble(
   double  value      // Value
   )
```

Parâmetros

value

[in] Variável para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteString

Escreve uma variável para arquivo do tipo string.

```
uint  WriteString(
   const string  value      // Value
   )
```

Parâmetros

value

[in] String para escrever.

Valor de retorno

Número de bytes escritos.

WriteString

Escreve uma variável para arquivo do tipo string.

```
uint  WriteString(
   const string  value,     // Value
   int           size       // Size
   )
```

Parâmetros

value

[in] String para escrever.

size

[in] Número de bytes para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteCharArray

Escreve um array de variável para arquivo do tipo char ou uchar.

```
uint  WriteCharArray(
   char&  array[],            // Array reference
   int    start_item=0,       // Start element
   int    items_count=-1      // Number of elements
   )
```

Parâmetros

array[]

[in] Array para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para o array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteShortArray

Escreve um array de variável para arquivo do tipo short ou ushort.

```
uint  WriteShortArray(
   short&  array[],            // Array to write
   int     start_item=0,       // Start element
   int     items_count=-1      // Number of elements to write
   )
```

Parâmetros

array[]

[in] Array para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para o array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteIntegerArray

Escreve um array de variável para arquivo do tipo int ou uint.

```
uint  WriteIntegerArray(
   int&  array[],            // Array to write
   int   start_item=0,       // Start element
   int   items_count=-1      // Number of elements to write
   )
```

Parâmetros

array[]

[in] Array para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para o array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteLongArray

Escreve um array de variável para arquivo do tipo long ou ulong.

```
uint  WriteLongArray(
   long&  array[],            // Array to write
   int    start_item=0,       // Start element
   int    items_count=-1      // Number of elements to write
   )
```

Parâmetros

array[]

[in] Array para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteFloatArray

Escreve um array de variável para arquivo do tipo float.

```
uint  WriteFloatArray(
   float&  array[],            // Array to write
   int     start_item=0,       // Start element
   int     items_count=-1      // Number of elements to write
   )
```

Parâmetros

array[]

[in] Array para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteDoubleArray

Escreve um array de variável para arquivo do tipo double.

```
uint  WriteDoubleArray(
   double&  array[],            // Array to write
   int      start_item=0,       // Start element
   int      items_count=-1      // Number of elements to write
   )
```

Parâmetros

array[]

[in] Matriz para escrever.

start_item=0

[in] Inicia a escrever a partir do elemento.

items_count=-1

[in] Número de elementos para escrever (-1 - para array completo).

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

WriteObject

Escreve dados da instância de classe herdeira CObject para arquivo.

```
bool  WriteObject(
   CObject*  object      // Reference to the object
   )
```

Parâmetros

object

[in] Referente à instância de classe herdeira CObject para escrever.

Valor de retorno

Verdadeiro se teve sucesso, falso se os dados não foram escritos.

© 2000-2024, MetaQuotes Ltd.

ReadChar

A partir do arquivo vai ler variável do tipo char ou uchar.

```
bool  ReadChar(
   char&  value      // Target variable
   )
```

Parâmetros

value

[in] Variável de destino do tipo char.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadShort

A partir do arquivo vai ler variável do tipo short ou ushort .

```
bool  ReadShort(
   short&  value
   )
```

Parâmetros

value

[in] Variável de destino do tipo short ou ushort.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadInteger

A partir do arquivo vai ler uma variável do tipo int ou uint .

```
bool  ReadInteger(
   int&  value      // Target variable
   )
```

Parâmetros

value

[in] Variável de destino do tipo int ou uint.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadLong

A partir do arquivo vai ler uma variável do tipo long ou ulong .

```
bool  ReadLong(
   long&  value
   )
```

Parâmetros

value

[in] Variável de destino do tipo long ou ulong.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadFloat

A partir do arquivo vai ler a variável do tipo float

```
bool  ReadFloat(
   float&  value      // Target variable
   )
```

Parâmetros

value

[in] Variável de destino do tipo float.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadDouble

A partir do arquivo vai ler uma variável do tipo double .

```
bool  ReadDouble(
   double&  value
   )
```

Parâmetros

value

[in] Variável de destino do tipo double.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadString

A partir do arquivo vai ler uma variável do tipo string.

```
bool  ReadString(
   string&  value      // Target string
   )
```

Parâmetros

value

[in] Variável de destino do tipo string.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

ReadString

A partir do arquivo vai ler uma variável do tipo string.

```
bool  ReadString(
   string&  value
   )
```

Parâmetros

value

[in] Variável de destino do tipo string.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadCharArray

A partir do arquivo vai ler um array da variável do tipo char or uchar.

```
bool  ReadCharArray(
   char&  array[],            // Target array
   int    start_item=0,       // Start element
   int    items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array de destino do tipo char ou uchar.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadShortArray

A partir do arquivo vai ler um array da variável do tipo short ou ushort .

```
bool  ReadShortArray(
   short&  array[],            // Target array
   int     start_item=0,       // Start element
   int     items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array de destino de tipo short ou ushort.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadIntegerArray

A partir do arquivo vai ler um array da variável do tipo int ou uint.

```
bool  ReadIntegerArray(
   int&  array[],            // Target array
   int   start_item=0,       // Start element
   int   items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array destino do tipo int ou uint.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadLongArray

A partir do arquivo vai ler um array da variável do tipo long ou ulong .

```
bool  ReadLongArray(
   long&  array[],            // Target array
   int    start_item=0,       // Start element
   int    items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array de destino do tipo long ou ulong.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadFloatArray

A partir do arquivo vai ler um array da variável do tipo float.

```
bool  ReadFloatArray(
   float&  array[],            // Target array
   int     start_item=0,       // Start element
   int     items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array de destino do tipo float.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadDoubleArray

A partir do arquivo vai ler um array da variável do tipo double.

```
bool  ReadDoubleArray(
   double&  array[],            // Target array
   int      start_item=0,       // Start element
   int      items_count=-1      // Number of elements to read
   )
```

Parâmetros

array[]

[in] Referente ao array de destino do tipo double.

start_item=0

[in] Elemento inicia a leitura.

items_count=-1

[in] Número de elementos para ler (-1 - lê até o final do arquivo).

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

ReadObject

A partir do arquivo vai ler dados da instância de classe herdeira CObject.

```
bool  ReadObject(
   CObject*  object      // Reference to the object
   )
```

Parâmetros

object

[in] Referente ao destino da instância de classe herdeira CObject para leitura.

Valor de retorno

verdadeiro se teve sucesso, falso se os dados não foram lidos.

© 2000-2024, MetaQuotes Ltd.

CFileTxt

CFileTxt é uma classe para o acesso simplificado aos arquivos de texto.

Descrição

Classe CFileTxt fornece um acesso aos arquivos de texto.

Declaração

```
class CFileTxt: public CFile
```

Título

```
#include <Files\FileTxt.mqh>
```

Hierarquia de herança

CObject
  CFile
    CFileTxt

Métodos de classe

Abrir métodos	
Open	Abre um arquivo de texto
Métodos de gravação	
WriteString	Escreve variável do tipo string para arquivo
Lendo métodos	
ReadString	A partir do arquivo vai ler variável do tipo string

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CFile

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2024, MetaQuotes Ltd.

Open

Se com sucesso, abre o arquivo de texto determinado e o atribui à instância de classe.

```
int  Open(
   const string  file_name,      // file name
   int           flags           // flags
   )
```

Parâmetros

file_name

[in] O nome do arquivo para abrir.

flags

[in] Flags de arquivo aberto (existe uma configuração forçada do flag FILE_TXT).

Valor de retorno

Identificador de arquivo aberto.

© 2000-2024, MetaQuotes Ltd.

WriteString

Escreve variável do tipo string para arquivo.

```
uint  WriteString(
   const string  value      // String to write
   )
```

Parâmetros

value

[in] String para escrever.

Valor de retorno

Número de bytes escritos.

© 2000-2024, MetaQuotes Ltd.

ReadString

A partir do arquivo vai ler uma variável do tipo string.

```
string  ReadString()
```

Valor de retorno

String que foram lidas.

© 2000-2024, MetaQuotes Ltd.

Operações String

Esta seção contém os detalhes técnicos das classes de operações string e descrições dos componentes correspondentes da biblioteca padrão MQL5.

O uso de classes de operações string vai poupar tempo no desenvolvimento de aplicações que utilizam operações de processamento de texto.

A Biblioteca Padrão MQL5 é alocada no diretório de trabalho do terminal, na pasta Include\Strings.

Classe	Descrição
CString	Classe para operações string

© 2000-2024, MetaQuotes Ltd.

CString

CString é uma classe para o acesso simplificado às variáveis do tipo string.

Descrição

A classe CFile proporciona o acesso simplificado a todos os seus descendentes para funções string do API MQL5.

Declaração

```
class CString: public CObject
```

Título

```
#include <Strings\String.mqh>
```

Hierarquia de herança

CObject
  CString

Métodos de classe

Métodos de acesso aos dados	
Str	Obtém uma string
Len	Obtém comprimento de uma string
Copy	Copia uma string
Métodos para preencher	
Fill	Preenche uma string com uma char específica
Assign	Atribui uma string
Append	Acrescenta uma string
Insert	Insere uma string
Métodos para comparar	
Compare	Compara uma string
CompareNoCase	Compara strings maiúscula e minúscula
Métodos substring	
Left	Obtém um número determinado de caracteres a partir do lado esquerdo de uma string

© 2000-2024, MetaQuotes Ltd.

Métodos de acesso aos dados	
Right	Obtém um número determinado de caracteres a partir do lado direito de uma string
Mid	Obtém um número determinado de caracteres a partir de uma string
Métodos para podar/excluir	
Trim	Remove todas as ocorrências a esquerda e a direita de uma configuração específica de caracteres a partir de uma string
TrimLeft	Remove todas as ocorrências a esquerda de uma configuração específica de caracteres a partir de uma string
TrimRight	Remove todas as ocorrências a direita de uma configuração específica de caracteres a partir de uma string
Clear	Limpa uma string
Métodos de converter	
ToUpper	Converte uma string para maiúsculas.
ToLower	Converte uma string para minúsculas.
Reverse	Reverte uma string
Os métodos de pesquisa	
Find	Procura a primeira combinação de uma substring
FindRev	Procura a última combinação de uma substring
Remover	Exclui uma substring de uma string
Replace	Substitui uma substring

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type

© 2000-2024, MetaQuotes Ltd.

Str

Obtém uma string.

```
string  Str() const;
```

Valor de retorno

Cópia de um string.

© 2000-2024, MetaQuotes Ltd.

Len

Obtém comprimento de uma string.

```
uint  Len() const;
```

Valor de retorno

Comprimento de uma string.

© 2000-2024, MetaQuotes Ltd.

Copy

Copia uma string pela referência.

```
void  Copy(
   string&  copy      // Reference
   ) const;
```

Parâmetros

copy

[in] Referente a uma string para copiar.

Copy

Copia uma string à instância de classe CString.

```
void  Copy(
   CString*  copy      // Object descriptor
   ) const;
```

Parâmetros

copy

[in] Descritor de objeto da classe CString.

© 2000-2024, MetaQuotes Ltd.

Fill

Preenche uma string com um char específico.

```
bool  Fill(
   short  character      // Character
   )
```

Parâmetros

character

[in] Caracteres para preenchimento.

Valor de retorno

verdadeiro se teve sucesso, falso se uma string não foi preenchida.

© 2000-2024, MetaQuotes Ltd.

Assign

Atribui uma string.

```
void  Assign(
   const string  str      // String to assign
   )
```

Parâmetros

str

[in] String para atribuir.

Assign

Atribui uma string à instância de classe CString.

```
void  Assign(
   CString*  str      // Object descriptor
   )
```

Parâmetros

str

[in] Descritor de objeto da classe CString para atribuir.

© 2000-2024, MetaQuotes Ltd.

Append

Acrescenta uma seqüência de caracteres.

```
void  Append(
   const string  str      // String to append
   )
```

Parâmetros

str

[in] String para anexar.

Append

Acrescenta uma seqüência de caracteres para a instância de classe CString.

```
void  Append(
   CString*  string      // Object descriptor
   )
```

Parâmetros

string

[in] Descritor de objeto da classe CString para anexar.

© 2000-2024, MetaQuotes Ltd.

Insert

Insere uma string de caracteres para uma posição específica.

```
uint  Insert(
   uint          pos,      // Position
   const string  str       // String to insert
   )
```

Parâmetros

pos

[in] Inserir posição.

str

[in] Inserir String.

Valor de retorno

Resultado do comprimento da string.

Insert

Insere uma string de caracteres na posição determinada para a instância de classe CString.

```
uint  Insert(
   uint      pos,      // Position
   CString*  str       // Object descriptor
   )
```

Parâmetros

pos

[in] Inserir posição.

str

[in] Descritor de objeto da classe CString para inserir.

Valor de retorno

Resultado do comprimento da string.

© 2000-2024, MetaQuotes Ltd.

Compare

Compara uma string.

```
int  Compare(
   const string  str      // String to compare
   ) const;
```

Parâmetros

str

[in] String para comparar.

Valor de retorno

Retorna 0 se as strings são iguais, -1 se uma string da classe é menor do que uma string para comparar, -1 se a string da classe é maior do que uma string para comparar.

Compare

Compara uma string com uma string de instância da classe CString.

```
int  Compare(
   CString*  str      // Object descriptor
   ) const;
```

Parâmetros

str

[in] descritor do objeto da classe CString para comparar.

Valor de retorno

Retorna 0 se as strings são iguais, -1 se uma string da classe é menor do que uma string para comparar, -1 se a string da classe é maior do que uma string para comparar.

© 2000-2024, MetaQuotes Ltd.

CompareNoCase

Compara string maiúsculas e minúscula.

```
int  CompareNoCase(
   const string  str      // String to compare
   ) const;
```

Parâmetros

str

[in] String para comparar.

Valor de retorno

Retorna 0 se as strings são iguais, -1 se uma string da classe é menor do que uma string para comparar, -1 se a string da classe é maior do que uma string para comparar.

CompareNoCase

Compara string (maiúscula e minúscula)com uma string de instância da classe CString.

```
int  CompareNoCase(
   CString*  str      // Object descriptor
   ) const;
```

Parâmetros

str

[in] descritor do objeto da classe CString para comparar.

Valor de retorno

Retorna 0 se as strings são iguais, -1 se uma string da classe é menor do que uma string para comparar, -1 se a string da classe é maior do que uma string para comparar.

© 2000-2024, MetaQuotes Ltd.

Left

Obtém um número determinado de caracteres a partir do lado esquerdo de uma string.

```
string  Left(
   uint  count      // Number of characters
   )
```

Parâmetros

count

[in] Número de caracteres.

Valor de retorno

Resultado da Substring.

© 2000-2024, MetaQuotes Ltd.

Right

Obtém um número determinado de caracteres a partir do lado direito de uma string.

```
string  Right(
   uint  count      // Number of characters
   )
```

Parâmetros

count

[in] Número de caracteres.

Valor de retorno

Resultado da Substring.

© 2000-2024, MetaQuotes Ltd.

Mid

Obtém um número determinado de caracteres a partir de uma string.

```
string  Mid(
   uint  pos,        // Position
   uint  count       // Number of characters
   )
```

Parâmetros

pos

[in] Posição de uma string.

count

[in] Número de caracteres.

Valor de retorno

Resultado da Substring.

© 2000-2024, MetaQuotes Ltd.

Trim

Remove todas as ocorrências a esquerda e a direita de uma configuração específica de caracteres (e ' ','\t','\r','\n') a partir de uma string.

```
int  Trim(
   const string  targets      // Set of characters to remove
   )
```

Parâmetros

targets

[in] Define os caracteres para remover.

Valor de retorno

Número de caracteres removidos.

Exemplo

```
//--- example for CString::Trim
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Source string '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Result string '%s'",str.Str());
  }
```

© 2000-2024, MetaQuotes Ltd.

TrimLeft

Remove todas as ocorrências a esquerda de uma configuração específica de caracteres (e ' ','\t','\r','\n') a partir de uma string.

```
int  TrimLeft(
   const string  targets      // Set of characters to remove
   )
```

Parâmetros

targets

[in] Define os caracteres para remover.

Valor de retorno

Número de caracteres removidos.

© 2000-2024, MetaQuotes Ltd.

TrimRight

Remove todas as ocorrências a direita de uma configuração específica de caracteres (e ' ','\t','\r','\n') a partir de uma string.

```
int  TrimRight(
   const string  targets      // Set of characters to remove
   )
```

Parâmetros

targets

[in] Define os caracteres para remover.

Valor de retorno

Número de caracteres removidos.

© 2000-2024, MetaQuotes Ltd.

Clear

Limpa uma string.

```
bool  Clear()
```

Valor de retorno

verdadeiro se teve sucesso, falso se uma string não foi limpa.

© 2000-2024, MetaQuotes Ltd.

ToUpper

Converte uma string para maiúsculas.

```
bool ToUpper()
```

Valor de retorno

verdadeiro se teve sucesso, falso se uma seqüência de caracteres não foi convertida.

© 2000-2024, MetaQuotes Ltd.

ToLower

Converte uma string para minúsculas.

```
bool  ToLower()
```

Valor de retorno

verdadeiro se teve sucesso, falso se uma seqüência de caracteres não foi convertida.

© 2000-2024, MetaQuotes Ltd.

Reverse

Reversão de uma string.

```
void  Reverse()
```

© 2000-2024, MetaQuotes Ltd.

Find

Procura a primeira combinação de uma substring.

```
int  Find(
   uint          start,          // Position
   const string  substring       // Substring to search for
   ) const;
```

Parâmetros

start

[in] O índice do caractere na string para iniciar a pesquisa com, ou 0 para começar a partir do início.

substring

[in] Substring para procurar.

Valor de retorno

O índice do primeiro caractere que corresponde a substring solicitado; -1 Se a substring não foi encontrada.

© 2000-2024, MetaQuotes Ltd.

FindRev

Procura a última combinação de uma substring.

```
int  FindRev(
   const string  substring      // Substring
   ) const;
```

Parâmetros

substring

[in] A substring para procurar.

Valor de retorno

O índice do último caractere que corresponde a substring solicitada; -1 Se a substring não foi encontrada.

© 2000-2024, MetaQuotes Ltd.

Remover

Exclui uma substring de uma string.

```
uint  Remove(
   const string  substring      // Substring to remove
   )
```

Parâmetros

substring

[in] A substring para procurar.

Valor de retorno

Número de substrings excluídas.

© 2000-2024, MetaQuotes Ltd.

Replace

Substitui uma substring a partir de uma string.

```
uint  Replace(
   const string  substring,      // Substring to replace
   const string  newstring       // New substring
   )
```

Parâmetros

substring

[in] A substring para procurar.

newstring

[in] Uma nova substring para substituir.

Valor de retorno

Número de substrings substituídas.

© 2000-2024, MetaQuotes Ltd.

Objetos Gráficos

Esta seção contém os detalhes técnicos de como trabalhar com classes de objetos gráficos e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

O uso de classes de objetos gráficos vai economizar tempo ao criar programas personalizados (scripts, expert).

Biblioteca Padrão MQL5 (em termos de objetos gráficos) é colocado no diretório de trabalho do terminal na pasta Include\ChartObjects.

Class/Group	Descrição
Base class for graphical object CChartObject	Classe base de um objeto gráfico
Lines	Classe do grupo "Lines"
Canais	Classe do grupo "Channels"
Gann Tools	Classe do grupo "Gann"
Fibonacci	Classe no grupo "Fibonacci"
Elliott Tools	Classe no grupo "Elliott"
Shapes	Classe no grupo "Shapes"
Arrows	Classe do grupo "Arrows"
Controls	Classe do grupo "Controls"

© 2000-2024, MetaQuotes Ltd.

CChartObject

CChartObject é uma classe base de objetos gráficos de tipo gráfico da biblioteca MQL5 padrão.

Descrição

A classe CChartObject proporciona o acesso simplificado para todos os seus descendentes para funções API MQL5.

Declaração

```
class CChartObject : public CObject
```

Título

```
#include <ChartObjects\ChartObject.mqh>
```

Hierarquia de herança

CObject

CChartObject

Descendentes diretos

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

Métodos de classe

Atributos	
ChartId	Obtém a ID que pertence a um gráfico
Window	Obtém o número de janelas do gráfico
Name	obtém/define o nome de um objeto gráfico
NumPoints	Obtém o número de pontos de ancoragem
Assign	
Attach	Vincula um gráfico
SetPoint	Define o ponto de ancoragem
Delete	
Delete	Exclui um gráfico
Detach	Destacar um gráfico
Shift	
ShiftObject	O movimento relativo do objeto

© 2000-2024, MetaQuotes Ltd.

Atributos	
ShiftPoint	O movimento relativo do ponto do objeto
Propriedades do objeto	
Time	Obtém/define as coordenadas de tempo do ponto de objeto
Price	Obtém/define a coordenada do preço do ponto de objeto
Color	obtém/define a cor do objeto
Style	Obtém/define o estilo de linha do objeto
Width	Obtém/define a largura do objeto de linha
BackGround	Obtém/define o flag desenhando o objeto em segundo plano
Selected	Obtém/define o flag "selecionada" de um objeto gráfico
Selectable	Obtém/define o flag selecionando o objeto
Descrição	Obtém/define o texto do objeto
Tooltip	Obtém/define a dica do objeto
Timeframes	Obtém/define a máscara dos flags para visualização do objeto
Z_Order	Obtém/define a prioridade de clicar num gráfico
CreateTime	Obtém a criação do objeto tempo
Níveis de propriedades do objeto	
LevelsCount	Obtém/define o número de níveis de objeto
LevelColor	Obtém/define a cor do nível de linha
LevelStyle	Obtém/define o nivel de estilo de linha
LevelWidth	Obtém/define a largura do nível de linha
LevelValue	Obtém/define o nível
LevelDescription	Obtém/define o nível do texto
Acesso a funções API MQL5	
GetInteger	Obtém o valor das propriedades de objetos
SetInteger	Define as propriedades do objeto
GetDouble	Obtém o valor das propriedades de objetos
SetDouble	Define as propriedades do objeto
GetString	Obtém o valor das propriedades de objetos
SetString	Define as propriedades do objeto
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
virtual Save	Método de entrada virtual no arquivo
virtual Load	Método virtual de leitura a partir de um arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

ChartId

Obtém a ID do gráfico pertencente a um objeto gráfico.

```
long  ChartId() const
```

Valor do Retorno

A Id do gráfico onde o objeto gráfico está localizado. Se o objeto não for encontrado, ele retorna -1.

Exemplo

```
//--- example for CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get chart idintifier of chart object
   long chart_id=object.ChartId();
  }
```

© 2000-2024, MetaQuotes Ltd.

Window

Obtém o número de janelas em que o gráfico é um objeto gráfico.

```
int  Window() const
```

Valor do Retorno

Número de janelas onde o gráfico é um objeto gráfico (0 - janela principal). Se o objeto não for encontrado, ele retorna -1.

Exemplo

```
//--- example for CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get window of chart object
   int window=object.Window();
  }
```

© 2000-2024, MetaQuotes Ltd.

Name (Método Get)

Obtém o nome do objeto gráfico.

```
string  Name() const
```

Valor do Retorno

Nome do objeto gráfico anexado a uma instância de classe. Se o objeto não for encontrado, retorna NULL.

Name (Método Set)

Define o nome do objeto gráfico.

```
bool  Name(
   string  name       // new name
   )
```

Parâmetros

name

[in] O novo nome do objeto gráfico.

Valor do Retorno

verdadeiro se obteve êxito, falso - não é possível alterar o nome.

Exemplo

```
//--- example for CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get name of chart object
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- set name of chart object
     object.Name("MyChartObject");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

NumPoints

Obtém o número de pontos de ancoragem de um objeto gráfico.

```
int  NumPoints() const
```

Valor do Retorno

Número de pontos ligados a um objeto gráfico anexado a uma instância de classe. Se o objeto não está anexado, ele retorna 0.

Exemplo

```
//--- example for CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get points count of chart object
   int points=object.NumPoints();
  }
```

© 2000-2024, MetaQuotes Ltd.

Attach

Vincula um objeto gráfico a uma instância de classe.

```
bool  Attach(
   long    chart_id,      // ID Grafico
   string  name,          // Name of the object
   int     window,        // Chart window
   int     points         // Number of points
   )
```

Parâmetros

chart_id

[out] Identificador do Gráfico

name

[in] Nome do objeto gráfico.

window

[in] Número de janela do gráfico (0 - janela principal).

points

[in] Número de pontos de ancoragem do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode vincular o objeto.

Exemplo

```
//--- example for CChartObject::Attach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- attach chart object
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Object attach error");
      return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SetPoint

Define as novas coordenadas de ponto de ancoragem do objeto gráfico.

```
bool  SetPoint(
   int       point,          // Point number
   datetime  new_time,       // Time coordinate
   double    new_price       // Price coordinate
   )
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

new_time

[in] Novo valor para a coordenada de tempo do ponto de ancoragem específico.

new_price

[in] Novo valor da coordenada de preço do ponto de ancoragem específico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode alterar as coordenadas do ponto.

Exemplo

```
//--- example for CChartObject::SetPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- set point of chart object
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Delete

Remove um objeto gráfico anexado ao gráfico.

```
bool  Delete()
```

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode remover o objeto.

Exemplo

```
//--- example for CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- detach chart object
   if(!object.Delete())
     {
      printf("Object delete error");
      return;
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Detach

Destaca o objeto gráfico.

```
void  Detach()
```

Valor do Retorno

Nenhum.

Exemplo

```
//--- example for CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- detach chart object
   object.Detach();
  }
```

© 2000-2024, MetaQuotes Ltd.

ShiftObject

Move um objeto gráfico.

```
bool  ShiftObject(
   datetime  d_time,       // Increment of time coordinate
   double    d_price       // Increment of price coordinate
   )
```

Parâmetros

d_time

[in] Incremento da coordenada de tempo para todos os pontos de ancoragem.

d_price

[in] Incremento da coordenada de preço para todos os pontos de ancoragem.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode deslocar o objeto.

Exemplo

```
//--- example for CChartObject::ShiftObject
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- shift chart object
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2024, MetaQuotes Ltd.

ShiftPoint

Desloca um ponto de ancoragem específico do objeto gráfico.

```
bool  ShiftPoint(
   int       point,       // Point number
   datetime  d_time,      // Increment of time coordinate
   double    d_price      // Increment of price coordinate
   )
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

d_time

[in] Incremento da coordenada de tempo para todos os pontos de ancoragem.

d_price

[in] Incremento da coordenada de preço para todos os pontos de ancoragem.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode deslocar o ponto.

Exemplo

```
//--- example for CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- shift point of chart object
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Time (Método Get)

Obtém a coordenada de tempo do ponto de ancoragem específico do objeto gráfico.

```
datetime  Time(
   int  point      // Point number
   ) const
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

Valor do Retorno

Coordenada de tempo de ponto de ancoragem específico do objeto gráfico anexado a uma instância de classe. Se o objeto não é anexado ou o objeto não tem este ponto, retorna 0.

Time (Método Set)

Define a cooordenada de tempo do ponto de ancoragem esepcífico de um objeto gráfico.

```
bool  Time(
   int       point,         // Point number
   datetime  new_time       // Time
   )
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

new_time

[in] Novo valor para a coordenada de tempo do ponto de ancoragem específico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode mudar as coordenadas de tempo.

Exemplo

```
//--- example for CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- get time of point chart object
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set time of point chart object
         object.Time(i,TimeCurrent());
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

Price (Método Get)

Obtém a coordenada de preço do ponto de ancoragem do objeto gráfico.

```
double  Price(
   int  point      // Point number
   ) const
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

Valor do Retorno

Coordenada de preço do ponto de ancoragem específico do objeto gráfico anexado a uma instância da classe. Se não houver nenhum objeto anexado ou o objeto não tem este ponto, ele retorna EMPTY_VALUE.

Price (Método Set)

Define a coordenada de preço do ponto de ancoragem específico de um objecto gráfico.

```
bool  Price(
   int     point,          // Point number
   double  new_price       // Price
   )
```

Parâmetros

point

[in] Número de um ponto de ancoragem.

new_price

[in] O novo valor para a coordenada de preço do ponto de ancoragem específico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - se não pode alterar a coordenada de preço.

Exemplo

```
//--- example for CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- get price of point chart object
```

© 2000-2024, MetaQuotes Ltd.

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- set price of point chart object
         object.Price(i,price);
        }
    }
  }
```

© 2000-2024, MetaQuotes Ltd.

Color (Método Get)

Obtém a cor da linha do objeto gráfico.

```
color  Color() const
```

Valor do Retorno

A cor da linha do objeto gráfico, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna CLR_NONE.

Color (Método Set)

Define a cor da linha do objeto gráfico.

```
bool  Color(
   color  new_color      // New color
   )
```

Parâmetros

new_color

[in] O novo valor de cor da linha do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode mudar a cor.

Exemplo

```
//--- example for CChartObject::Color
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get color of chart object
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- set color of chart object
     object.Color(clrRed);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Style (Método Get)

Obtém o estilo de linha do objeto gráfico.

```
ENUM_LINE_STYLE  Style() const
```

Valor do Retorno

Estilo de linha do objeto gráfico anexado à instância de classe. Se não houver nenhum objeto anexado, retorna WRONG_VALUE.

Style (Método Set)

Define o estilo de linha do objeto gráfico.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // Style
   )
```

Parâmetros

new_style

[in] Novo valor do estilo de linha.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o estilo.

Exemplo

```
//--- example for CChartObject::Style
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get style of chart object
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- set style of chart object
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Width (Método Get)

Obtém a linha do objeto gráfico.

```
int  Width() const
```

Valor do Retorno

A linha do objeto gráfico que esteja vinculada a uma instância de classe. Se não está anexado ao objeto, retorna -1.

Width (Método Set)

Define a linha do objeto gráfico.

```
bool  Width(
   int  new_width      // Thickness
   )
```

Parâmetros

new_width

[in] O novo valor da linha do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - se não pode alterar a largura.

Exemplo

```
//--- example for CChartObject::Width
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get width of chart object
   int width=object.Width();
   if(width!=1)
     {
      //--- set width of chart object
      object.Width(1);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Background (Método Get)

Obtém o flag desenhando o objeto em segundo plano (fundo).

```
bool  Background() const
```

Valor do Retorno

Flag desenhado em segundo plano, um objeto gráfico que esteja vinculado a uma instância de classe. Se não vinculado, o objeto retorna falso.

Background (Método Set)

Define o flag desenhando o objeto em segundo plano (fundo).

```
bool  Background(
   bool  background      // Value of the flag
   )
```

Parâmetros

background

[in] O novo valor do flag desenhando um objeto em segundo plano (fundo).

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o flag.

Exemplo

```
//--- example for CChartObject::Background
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get background flag of chart object
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- set background flag of chart object
     object.Background(true);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Selected (Método Get)

Obtém o flag indicando que o objeto gráfico foi selecionado Em outras palavras - se o objeto gráfico é selecionado ou não.

```
bool  Selected() const
```

Valor do Retorno

O estado em que o objeto anexado a uma instância de classe é selecionado. Se não houver nenhum objeto anexado, retorna falso.

Selected (Método Set)

Define o flag indicando que o objeto gráfico foi selecionado.

```
bool  Selected(
   bool  selected      // Value of the flag
   )
```

Parâmetros

selected

[in] Novo valor do flag.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o flag.

Exemplo

```
//--- example for CChartObject::Selected
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get the "selected" flag of chart object
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- set the "selected" flag of chart object
     object.Selected(false);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Selectable (Método Get)

Obtém o flag que indica a capacidade de um objeto gráfico ser selecionado. Em outras palavras - se o objeto gráfico pode ser selecionada ou não.

```
bool  Selectable() const
```

Valor do Retorno

Flag indicando a capacidade do objeto ser selecionado, anexado a uma instância da classe. Se não houver nenhum objeto anexado, retorna falso.

Selectable (Método Set)

Define o flag que indica a capacidade de um objeto gráfico a ser selecionado.

```
bool  Selectable(
   bool  selectable      // Value of the flag
   )
```

Parâmetros

selectable

[in] Novo valor do flag.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o flag.

Exemplo

```
//--- example for CChartObject::Selectable
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get the "selectable" flag of chart object
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- set the "selectable" flag of chart object
     object.Selectable(false);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Description (Método Get)

Obtém uma descrição (texto) do objeto gráfico.

```
string  Description() const
```

Valor do Retorno

Descrição (texto) do objeto gráfico que esteja vinculado a uma instância de classe. Se nenhum objeto é vinculado, retorna NULL.

Description (Método Set)

Define a descrição(texto) do objeto gráfico.

```
bool  Description(
   string  text      // Text
   )
```

Parâmetros

text

[in] Novo valor das descrições (texto) do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode alterar a descrição (texto).

Exemplo

```
//--- example for CChartObject::Description
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get description of chart object
   string description=object.Description();
   if(description=="")
     {
      //--- set description of chart object
      object.Description("MyObject");
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Tooltip (Método Get)

Obtém o texto da dica de ferramenta do objeto gráfico.

```
string  Tooltip() const
```

Valor de retorno

O texto de uma dica de ferramenta do objeto gráfico anexado a uma instância da classe. Se o objeto não foi anexado, ele retorna NULL.

Tooltip (Método Set)

Define o texto da dica de ferramenta de um objeto gráfico.

```
bool  Tooltip(
   string  new_tooltip      // new text of a tooltip
   )
```

Parâmetros

new_tooltip

[in] O novo texto de uma dica de ferramenta.

Valor de retorno

verdadeiro - se bem sucedido, falso se a dica de ferramenta falhar.

Nota:

Se a propriedade não é definida, então a dica de ferramenta é gerada automaticamente pelo terminal e será exibida. Uma dica de ferramenta pode ser desativada, definindo o valor "\n" (linha de alimentação).

© 2000-2024, MetaQuotes Ltd.

Timeframes (Método Get)

Obtém o flag de visibilidade do objeto gráfico.

```
int  Timeframes() const
```

Valor do Retorno

Flags de visibilidade do objeto gráfico anexadas a uma instância da classe. Se o objeto não é anexado, ele retorna 0.

Timeframes (Método Set)

Define o flag de visibilidade do objeto gráfico.

```
bool  Timeframes(
   int  new_timeframes      // Visibility flags
   )
```

Parâmetros

new_timeframes

[in] Novos flags de visibilidade do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode mudar as flags de visibilidade.

Exemplo

```
//--- example for CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get timeframes of chart object
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- set timeframes of chart object
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Z_Order (Método Get)

Obtém o valor da prioridade do objeto gráfico ao clicar no gráfico (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Valor do Retorno

Prioridade de um objeto gráfico, anexado à instância de classe. Se não houver nenhum objeto anexado, ele retorna 0.

Z_Order (Método Set)

Define a prioridade do objeto gráfico ao clicar no gráfico (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // new priority
   )
```

Parâmetros

value

[in] O novo valor da prioridade do objeto gráfico ao clicar no gráfico (CHARTEVENT_CLICK).

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode alterar a prioridade.

Observação

Z_Order é uma prioridade de um objeto gráfico para receber eventos ao clicar no gráfico (CHARTEVENT_CLICK). Ao definir o valor maior do que 0 (valor padrão), você pode aumentar a prioridade objeto.

© 2000-2024, MetaQuotes Ltd.

CreateTime

Obtém o tempo para criar o objeto gráfico.

```
datetime  CreateTime() const
```

Valor do Retorno

Tempo para criar objeto gráfico que esteja vinculado a uma instância de classe. Se o objeto não é vinculado, ele retorna 0.

Exemplo

```
//--- example for CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get create time of chart object
   datetime create_time=object.CreateTime();
  }
```

© 2000-2024, MetaQuotes Ltd.

LevelsCount (Método Get)

Obtém o número de níveis do objeto gráfico.

```
int  LevelsCount() const
```

Valor do Retorno

Número de níveis do objto gráfico anexado a uma instância de classe. Se o objeto não é anexado, ele retorna 0.

LevelsCount (Método Set)

Define o número de níveis do objeto gráfico.

```
bool  LevelsCount(
   int  levels      // Number of levels
   )
```

Parâmetros

levels

[in] O novo número de níveis do objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível alterar o número de níveis.

Exemplo

```
//--- example for CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get levels count of chart object
   int levels_count=object.LevelsCount();
   //--- set levels count of chart object
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2024, MetaQuotes Ltd.

LevelColor (Método Get)

Obtém a cor da linha de nível específico do objeto gráfico.

```
color  LevelColor(
   int  level      // Level number
   ) const
```

Parâmetros

level

[in] Número de nível.

Valor do Retorno

Cor da linha de nível específico do objeto gráfico anexado a uma instância de classe. Se não está vinculado ao objeto ou não existe um nível especificado de objeto, retorna CLR_NONE.

LevelColor (Método Set)

Define a cor da linha de nível específico do objeto gráfico.

```
bool  LevelColor(
   int    level,          // Level number
   color  new_color       // New color
   )
```

Parâmetros

level

[in] Número de nível.

new_color

[in] Nova cor de linha de nível específico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode mudar a cor.

Exemplo

```
//--- example for CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get level color of chart object
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set level color of chart object
         object.LevelColor(i,clrRed);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelStyle (Método Get)

Obtém o estilo da linha de nível específico do objeto gráfico.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // Level number
   ) const
```

Parâmetros

level

[in] Número de nível.

Valor do Retorno

Estilo da linha de nível específico do objeto gráfico anexado a uma instância de classe. Se não é anexado ao objeto ou o objeto não tem um nível específico, retorna WRONG_VALUE.

LevelStyle (Método Set)

Define estilo da linha de nível específico do objeto gráfico.

```
int  LevelStyle(
   int               level,      // Level number
   ENUM_LINE_STYLE   style       // Line Style
   )
```

Parâmetros

level

[in] Número de nível.

style

[in] Novo estilo de linha do nível especificado.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o estilo.

Exemplo

```
//--- example for CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get level style of chart object
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set level style of chart object
         object.LevelStyle(i,STYLE_SOLID);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelWidth (Get Method)

Obtém a largura da linha de nível específico do objeto gráfico.

```
int  LevelWidth(
   int  level      // Level number
   ) const
```

Parâmetros

level

[in] Número de nível.

Valor do Retorno

A largura da linha de nível específico do objeto gráfico anexado à instância de classe. Se não houver nenhum objeto anexado ou o objeto não tem o nível especificado, ele retorna -1.

LevelWidth (Método Set)

Define a largura da linha de nível específico do objeto gráfico.

```
bool  LevelWidth(
   int  level,          // Level number
   int  new_width       // New width
   )
```

Parâmetros

level

[in] Número de nível.

new_width

[in] Nova largura da linha do nível especificado.

Valor do Retorno

verdadeiro - se bem sucedido, falso - se não pode alterar a largura.

Exemplo

```
//--- example for CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get level width of chart object
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set level width of chart object
         object.LevelWidth(i,1);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelValue (Método Get)

Obtém o valor de nível específico do objeto gráfico.

```
double  LevelValue(
   int  level      // Level number
   ) const
```

Parâmetros

level

[in] Número de nível.

Valor do Retorno

O valor deste nível de objeto gráfico que está vinculado a uma instância de classe. Se não é vinculado ao objeto ou não existe um nível específico do objeto, retorna EMPTY_VALUE.

LevelValue (Método Set)

Define o valor do nível específico do objeto gráfico.

```
bool  LevelValue(
   int     level,          // Level number
   double  new_value       // New value
   )
```

Parâmetros

level

[in] Número de nível.

new_value

[in] Novo valor do nível especificado.

Valor do Retorno

veradeiro - sem bem sucedido, falso - se não pode alterar o valor

Exemplo

```
//--- example for CChartObject::LevelValue
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get level value of chart object
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set level value of chart object
         object.LevelValue(i,0.1*i);
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

LevelDescription (Método Get)

Obtém uma descrição (texto) do nível de objeto gráfico.

```
string  LevelDescription(
   int  level       // Level number
   ) const
```

Parâmetros

level

[in] Número de níveis de objeto gráfico.

Valor do Retorno

Descrição (texto) do nível de objeto gráfico que esteja vinculado a uma instância de classe. Se não vincular o objeto ou não é especificado o nível do objeto, retorna NULL.

LevelDescription (Método Set)

Define a descrição (texto) do nível de objeto gráfico.

```
bool  LevelDescription(
   int     level ,      // Level number
   string  text         // Text
   )
```

Parâmetros

level

[in] Número de nível de objeto gráfico.

text

[in] O novo valor da descrição (texto) do nível de objeto gráfico.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode alterar a descrição (texto).

Exemplo

```
//--- example for CChartObject::LevelDescription
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get level description of chart object
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2024, MetaQuotes Ltd.

```
        {
         //--- set level description of chart object
         object.LevelDescription(i,"Level_"+IntegerToString(i));
        }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

GetInteger

Fornece acesso simplificado às funções ObjectGetInteger() do API MQL5 para as propriedades de valor tipo integer (bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, color) viculadas a uma instância da classe gráfica. Existem duas versões de uma chamada desta função:

Obtendo um valor de propriedade, sem verificar a exatidão

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // Identifier of integer-property
   int                           modifier=-1       // Modifier
   ) const
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo double.

modifier=-1

[in] Modificador (índexador) de recursos do tipo double.

Valor do Retorno

Se bem sucedido, retorna o valor da propriedade de tipo integer, se houver erro vai retornar 0.

Obtendo um valor de propriedade ao verificar a correção de tal tratamento

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // Identifier of integer-property
   int                           modifier,      // Modifier
   long&                         value          // Link to variable
   ) const
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo integer do objeto.

modifier

[in] Modificador (indexador) da propriedade tipo integer.

value

[out] Referência a uma variável para acomodar as propriedades de valor tipo integer.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não pode obter a propriedade tipo integer.

Exemplo

```
//--- example for CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
{
 CChartObject object;
 //--- get color of chart object by easy method
 printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
 //--- get color of chart object by classic method
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("Get integer property error %d",GetLastError());
    return;
   }
 else
    printf("Objects color is %s",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- get levels width by easy method
    printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- get levels width by classic method
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("Get integer property error %d",GetLastError());
       return;
      }
    else
       printf("Level %d width is %d",i,width_value);
   }
}
```

© 2000-2024, MetaQuotes Ltd.

SetInteger

Fornece acesso simplificado às funções ObjectSetInteger() do API MQL5 para alterar as propriedades tipo integer (bool, carvão animal, uchar, short, ushort, int, uint, long, ulong, datetime, cor) ligadas a uma instância de classe gráfica. Existem duas versões de uma chamada da função:

Define um valor de propriedade que não requer um modificador

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // Identifier of integer-property
   long                           value         // Value
   )
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo integer do objeto.

value

[in] novas propriedades mutáveis de valor tipo integer.

Define um valor de propriedade que indica o modificador

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,       // Identifier of integer-property
   int                            modifier,      // Modifier
   long                           value          // Value
   )
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo integer do objeto.

modifier

[in] Modificador (indexador) da propriedade tipo integer.

value

[in] novas propriedades mutáveis de valor tipo integer.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar a propriedade tipo integer.

Exemplo

```
//--- example for CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- set new color of chart object
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!object.SetInteger(OBJPROP_COLOR,clrRed))
    {
     printf("Set integer property error %d",GetLastError());
     return;
    }
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- set levels width
     if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
       {
        printf("Set integer property error %d",GetLastError());
        return;
       }
    }
 }
```

© 2000-2024, MetaQuotes Ltd.

GetDouble

Fornece acesso simplificado às funções ObjectGetDouble() do API MQL5 para obter os valores de propriedades duplas (tipo float e double) do objeto gráfico, atribuídos à instância de classe. Existem duas versões de uma chamada desta função:

Obtendo um valor de propriedade, sem verificar a exatidão

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // Identifier of double-property
   int                          modifier=-1       // Modifier
   ) const
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo double.

modifier=-1

[in] Modificador (índexador) de recursos do tipo double.

Valor do Retorno

Se for bem sucedido, retorna o valor da propriedade de tipo double, em caso de erro, ele retorna EMPTY_VALUE.

Obtendo um valor de propriedade ao verificar a correção de tal tratamento

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,       // Identifier of double-property
   int                          modifier,      // Modifier
   double&                      value          // Link to variable
   ) const
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo double.

modifier

[in] Modificador (índexador) de recursos do tipo double.

value

[out] Referência a uma variável para acomodar as propriedades de valor tipo double.

Valor do Retorno

verdadeiro - se bem sucedida, falso - não pode obter um recurso tipo double.

Exemplo

```
//--- example for CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
```

© 2000-2024, MetaQuotes Ltd.

```
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- get levels value by easy method
      printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
      //--- get levels value by classic method
      double value;
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
         printf("Get double property error %d",GetLastError());
         return;
        }
      else
         printf("Level %d value=%f",i,value);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

SetDouble

Fornece acesso simplificado às funções ObjectSetDouble() do API MQL5 para alterar as propriedades duplas (tipo float e double) vinculadas a uma instância do objeto gráfico da classe. Existem duas versões de uma chamada desta função:

Define um valor de propriedade que não requer um modificador

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // Identifier of double-property
   double                        value         // Value
   )
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo double.

value

[in] Novos valores mutáveis de recursos tipo double

Define um valor de propriedade que indica o modificador

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // Identifier of double-property
   int                           modifier,      // Modifier
   double                        value          // Value
   )
```

Parâmetros

prop_id

[in] ID de propriedades gráficas tipo double.

modifier

[in] Modificador (indexador) de propriedade do tipo double.

value

[in] Novos valores mutáveis de propriedade tipo double

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar o recurso tipo double.

Exemplo

```
//--- example for CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

© 2000-2024, MetaQuotes Ltd.

```
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- set level value of chart object
     if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
       {
        printf("Set double property error %d",GetLastError());
        return;
       }
    }
}
```

© 2000-2024, MetaQuotes Ltd.

GetString

Fornece acesso simplificado às funções ObjectGetString() do API MQL5 para as propriedades tipo string, é vinculado a uma instância de classe do objeto gráfico. Existem duas versões de uma chamada desta função:

Obtendo um valor de propriedade, sem verificar a exatidão

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // Identifier of string-property
   int                          modifier=-1       // Modifier
   ) const
```

Parâmetros

prop_id

[in] Identificador das propriedades tipo string do objeto gráfico.

modifier=-1

[in] Modificador (indexador) de propriedade tipo string.

Valor do Retorno

Valor da propriedade tipo string.

Obtendo um valor de propriedade ao verificar a correção de tal tratamento

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // Identifier of string-property
   int                          modifier,      // Modifier
   string&                      value          // Link to variable
   ) const
```

Parâmetros

prop_id

[in] Identificador das propriedades tipo string do objeto gráfico.

modifier

[in] Modificador (indexador) de propriedade tipo string.

value

[out] Referência a uma variável para acomodar as propriedades de valor tipo string.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível obter a propriedade tipo string.

Exemplo

```
//--- example for CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2024, MetaQuotes Ltd.

```
{
 CChartObject object;
 string       value;
 //--- get name of chart object by easy method
 printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
 //--- get name of chart object by classic method
 if(!object.GetString(OBJPROP_NAME,0,value))
   {
    printf("Get string property error %d",GetLastError());
    return;
   }
 else
    printf("Object name is '%s'",value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- get levels description by easy method
    printf("Level %d description is '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
    //--- get levels description by classic method
    if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
      {
       printf("Get string property error %d",GetLastError());
       return;
      }
    else
       printf("Level %d description is '%s'",i,value);
   }
}
```

© 2000-2024, MetaQuotes Ltd.

SetString

Fornece acesso simplificado às funções ObjectSetString() do API MQL5 para alterar as propriedades amarradas do tipo string à instância do objeto gráfico de classe. Existem duas versões de uma chamada desta função:

Define um valor de propriedade que não requer um modificador

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // Identifier of string-property
   string                        value         // Value
   )
```

Parâmetros

prop_id

[in] Identificador das propriedades tipo string do objeto gráfico.

value

[in] Novos valores mutáveis das propriedades do tipo string.

Define um valor de propriedade que indica o modificador

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,       // Identifier of string-property
   int                           modifier,      // Modifier
   string                        value          // Value
   )
```

Parâmetros

prop_id

[in] Identificador das propriedades tipo string do objeto gráfico.

modifier

[in] Modificador (indexador) de propriedade tipo string.

value

[in] Novos valores mutáveis das propriedades do tipo string.

Valor do Retorno

verdadeiro - se bem sucedido, falso - não é possível mudar propriedade tipo string.

Exemplo

```
//--- example for CChartObject::SetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- set new name of chart object
```

© 2000-2024, MetaQuotes Ltd.

```
  if(!object.SetString(OBJPROP_NAME,"MyObject"))
    {
     printf("Set string property error %d",GetLastError());
     return;
    }
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- set levels description
     if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
       {
        printf("Set string property error %d",GetLastError());
        return;
       }
    }
 }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva todos os parâmetros do objeto no arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo aberto anteriormente usando a função FileOpen(...).

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CChartObject::Save
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object=new CChartObject;
   //--- set object parameters
   //--- . . .
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega os parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo previamente aberto usando a função FileOpen(...).

Valor do Retorno

verdadeiro - concluído com sucesso, falso - com erro.

Exemplo

```
//--- example for CChartObject::Load
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CChartObject object;
   //--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- use object
   //--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Obtém o identificador de tipo do objeto gráfico .

```
virtual int  Type() const
```

Valor do Retorno

Identificador do tipo de objeto (0x8888 para CChartObject).

Exemplo

```
//--- example for CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- get object type
   int type=object.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Objetos de Linhas

Um grupo de objetos gráficos chamado "Lines".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Lines" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectVLine	Objeto gráfico "Vertical Line"
CChartObjectHLine	Objeto gráfico "Horizontal Line"
CChartObjectTrend	Objeto gráfico "Trend Line"
CChartObjectTrendByAngle	Objeto gráfico "Trend Line by Angle"
CChartObjectCycles	Objeto gráfico "Cyclic Lines"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectVLine

CChartObjectVLine é uma classe de acesso simplificado às propriedades do objeto gráfico "Vertical Line".

Descrição

Classe CChartObjectVLine fornece acesso às propriedades do objeto gráfico "Vertical Line".

Declaração

```
class CChartObjectVLine : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectVLine

Métodos de classe

Create	
Create	Cria o objeto gráfico "Vertical Line"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Vertical Line".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time           // Time coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time

[in] Coordenadas de tempo do ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_VLINE para CChartObjectVLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectHLine

CChartObjectHLine é uma classe de acesso simplificado às propriedade de objeto gráfico "Horizonal Line".

Descrição

Classe CChartObjectHLine fornece acesso às propriedade de objeto gráfico "Horizonal Line".

Declaração

```
class CChartObjectHLine : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hierarquia de herança

COBject
  CChartObject
    CChartObjectHLine

Métodos de classe

Create	
Create	Cria o objeto gráfico "Horizonal Line"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o gráfico objeto "Horizonal Line".

```
bool  Create(
   long    chart_id,      // Chart identifier
   string  name,          // Object name
   long    window,        // Chart window
   double  price          // Price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

price

[in] Coordenadas de preço do ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_HLINE para CChartObjectHLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTrend

CChartObjectTrend é uma classe para acesso simplificado às propriedades do objeto gráfico "Trend Line".

Descrição

Classe CChartObjectTrend fornece acesso a "às propriedades do objeto gráfico "Trend Line".

Declaração

```
class CChartObjectTrend : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend

Descendentes diretos

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

Métodos de classe

Create	
Create	Cria objeto gráfico "Trend Line"
Propriedades	
RayLeft	Obtém/Define a propriedade "Ray Left"
RayRight	Obtém/Define a propriedade "Ray Right"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Trend Line".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // 1st time coordinate
   double    price1,        // 1st price coordinate
   datetime  time2,         // 2nd time coordinate
   double    price2         // 2nd price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

RayLeft (Método Get)

Obtém o valor da Propriedade "Ray Left".

```
bool  RayLeft() const
```

Valor de retorno

O valor da propriedade "Ray Left", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

RayLeft (Método Set)

Define um novo valor de flag à propriedade "Ray Left".

```
bool  RayLeft(
   bool  ray      // flag
   )
```

Parâmetros

ray

[in] O novo valor da Propriedade Ray Left.

Valor de retorno

verdadeiro se obteve êxito, falso se a bandeira não mudou erro.

© 2000-2024, MetaQuotes Ltd.

RayRight (Método Get)

Obtém o valor da Propriedade "Ray Right".

```
bool  RayRight() const
```

Valor de retorno

O valor da propriedade "Ray Right", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

RayRight (Método Set)

Define um novo valor de flag à propriedade "Ray Right".

```
bool  RayRight(
   bool  ray      // flag
   )
```

Parâmetros

ray

[in] O novo valor da propriedade "Ray Right".

Valor de retorno

verdadeiro se obteve êxito, falso se a bandeira não mudou erro.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipula o arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] manipulador do arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_TREND para CChartObjectTrend).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTrendByAngle

CChartObjectTrendByAngle é uma classe de acesso simplificado às propriedades do objeto gráfico "Trend Line by Angle".

Descrição

Classe CChartObjectTrendByAngle fornece acesso propriedades do objeto gráfico "Trend Line by Angle".

Declaração

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle

Descendentes diretos

CChartObjectGannLine

Métodos de classe

Create	
Create	Cria o objeto gráfico "Trend Line by Angle"
Propriedades	
Angle	Obtém/Define a propriedade "Angle"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Trend Line by Angle"

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // 1st time coordinate
   double    price1,        // 1st price coordinate
   datetime  time2,         // 2nd time coordinate
   double    price2         // 2nd price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Angle (Método Get)

Obtém o valor da propriedade "Angle".

```
double  Angle() const
```

Valor de retorno

O valor da propriedade "Angle", atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

Angle (Método Set)

Sets new value for the "Angle" property.

```
bool  Angle(
   double  angle      // Angle
   )
```

Parâmetros

angle

[in] O novo valor da propriedade "Angle".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_TRENDBYANGLE para CChartObjectTrendByAngle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectCycles

CChartObjectCycles é uma classe de acesso simplificado às propriedades do objeto gráfico "Cyclic Lines".

Descrição

Classe CChartObjectCycles fornece acesso às propriedades do objeto gráfico "Cyclic Lines".

Declaração

```
class CChartObjectCycles : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectCycles

Métodos de classe

Create	
Create	Cria o objeto gráfico "Cyclic Lines"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Cyclic Lines"

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // 1st time coordinate
   double    price1,        // 1st price coordinate
   datetime  time2,         // 2nd time coordinate
   double    price2         // 2nd price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_CYCLES para CChartObjectCycles).

© 2000-2024, MetaQuotes Ltd.

Objetos de Canais

Um grupo de objetos gráficos chamado "Channels".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de classes de objetos gráficos chamado "Channels" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectChannel	Objeto gráfico "Equidistant Channel"
EquidistantCChartObjectRegression	Objeto gráfico "Linear Regression Channell"
CChartObjectStdDevChannel	Objeto gráfico "Standard deviations Channel"
CChartObjectPitchfork	Objeto gráfico "Andrew's Pitchfork"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectChannel

O CChartObjectChannel é uma classe para o acesso simplificado a propriedades de objetos gráficos "Equidistant Channel".

Descrição

A classe CChartObjectChannel fornece acesso as propriedades do objeto "Equidistant Channel".

Declaração

```
class CChartObjectChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

Métodos de classe

Create	
Create	Cria o objeto gráfico "Equidistant Channel"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Equidistant Channel"

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // Time coordinate of the first anchor point
   double    price1,        // Price coordinate of the first anchor point
   datetime  time2,         // Time coordinate of the second anchor point
   double    price2,        // Price coordinate of the second anchor point
   datetime  time3,         // Time coordinate of the third anchor point
   double    price3         // Price coordinate of the third anchor point
   )
```

Parâmetros

chart_id

[in] ID do gráfico(0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Tipo de identificador do objeto (OBJ_CHANNEL for CChartObjectChannel).

© 2000-2024, MetaQuotes Ltd.

EquidistantCChartObjectRegression

CChartObjectRegression é uma classe para acesso simplificado às propriedades do objeto gráfico "Linear Regression Channel".

Descrição

A classe CChartObjectRegression fornece acesso simplificado às propriedades do objeto gráfico "Linear Regression Channel".

Declaração

```
class CChartObjectRegression : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

Métodos de classe

Create	
Create	Cria o objeto gráfico "Linear Regression Channel"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Linear Regression Channel"

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // First time coordinate
   datetime  time2          // Second time coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Tipo de identificador do objeto (OBJ_REGRESSION for CChartObjectRegression).

© 2000-2024, MetaQuotes Ltd.

CChartObjectStdDevChannel

CChartObjectStdDevChannel é uma classe para o acesso simplificado às propriedades do objeto gráfico "Standard Deviation Channel".

Descrição

A classe CChartObjectStdDevChannel fornece acesso às propriedades do objeto "Standard Deviation Channel".

Declaração

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

Métodos de classe

Create	
Create	Cria o objeto gráfico "Standard Deviation Channel"
Propriedades	
Deviations	Obtém/Define a propriedade "Deviation"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Standard Deviation Channel"

```
bool  Create(
   long      chart_id,       // Chart identifier
   string    name,           // Object name
   int       window,         // Chart window
   datetime  time1,          // First time coordinate
   datetime  time2,          // Second time coordinate
   double    deviation       // Deviation
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

deviation

[in] O valor numérico para a propriedade "Deviation".

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Deviation (Método Get)

Obtém o valor numérico à propriedade "Deviation".

```
double  Deviation() const
```

Valor de retorno

O valor numérico da propriedade "Deviation", atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

Deviation (Método Set)

Define o valor numérico para a propriedade "Deviation" .

```
bool  Deviation(
   double  deviation      // Deviation
   )
```

Parâmetros

deviation

[in] O novo valor da propriedade "Deviation".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipula o arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipula o arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Tipo de identificador do objetoOBJ_STDDEVCHANNEL for CChartObjectStdDevChannel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectPitchfork

A CChartObjectPitchfork é uma classe para acesso simplificado das propriedades do objeto gráfico "Pitchfork de Andrew" .

Descrição

A classe CChartObjectPitchfork fornece o acesso das propriedades do objeto "Pitchfork de Andrew"

Declaração

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

Métodos de classe

Create	
Create	Cria o objeto gráfico "Pitchfork de Andrew "
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Pitchfork de Andrew "

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // Time coordinate of the first anchor point
   double    price1,        // Price coordinate of the first anchor point
   datetime  time2,         // Time coordinate of the second anchor point
   double    price2,        // Price coordinate of the second anchor point
   datetime  time3,         // Time coordinate of the third anchor point
   double    price3         // Price coordinate of the third anchor point
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Tipo de identificador do objeto (OBJ_PITCHFORK for CChartObjectPitchfork).

© 2000-2024, MetaQuotes Ltd.

Ferramentas Gann

Um grupo de objetos gráficos chamado "Gann Tools".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Gann Tools" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectGannLine	Objeto gráfico "Gann Line"
CChartObjectGannFan	Objeto gráfico "Gann Fan"
CChartObjectGannGrid	Objeto gráfico "Gann Grid"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannLine

CChartObjectGannLine é uma classe para acesso simplificado às propriedades do objeto gráfico "Gann Line".

Descrição

Classe CChartObjectGannLine fornece acesso às propriedades do objeto gráfico "Gann Line".

Declaração

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectTrendByAngle
        CChartObjectGannLine

Métodos de classe

Create	
Create	Cria o objeto gráfico "Gann Line"
Propriedades	
PipsPerBar	Obtém/Define a propriedade "Scale"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Métodos herdados da classe CChartObjectTrendByAngle

Angle, Angle, Create

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Gann Line".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    ppb            // Pips per bar
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

ppb

[in] Pips por barra.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtém o valor da propriedade "Pips per bar".

```
double  PipsPerBar() const
```

Valor de retorno

Valor da propriedade "Pips per bar" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

PipsPerBar (Método Set)

Define um novo valor à propriedade "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips per bar
   )
```

Parâmetros

ppb

[in] O novo valor da propriedade "Pips per bar".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_GANNLINE para CChartObjectGannLine).

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannFan

CChartObjectGannFan é uma classe para acesso simplificado às propriedades do objeto gráfico "Gann Fan".

Descrição

Classe CChartObjectGannFan fornece acesso às propriedades do objeto gráfico "Gann Fan"

Declaração

```
class CChartObjectGannFan : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

Métodos de classe

Create	
Create	Cria o objeto gráfico "Gann Fan"
Propriedades	
PipsPerBar	Obtém/Define a propriedade "Pips per bar"
Downtrend	Obtém/Define a propriedade "Downtrend"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Gann Fan".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    ppb            // Pips per bar
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

ppb

[in] Pips por barra.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtém o valor da propriedade "Pips per bar".

```
double  PipsPerBar() const
```

Valor de retorno

Valor da propriedade "Pips per bar" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

PipsPerBar (Método Set)

Define um novo valor à propriedade "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips per bar
   )
```

Parâmetros

ppb

[in] O novo valor da propriedade "Pips per bar".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Downtrend (Método Get)

Obtém o valor da propriedade "Downtrend".

```
bool  Downtrend() const
```

Valor de retorno

Valor da propriedade do objeto "Downtrend", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Downtrend (Método Set)

Define novo valor da propriedade "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Flag value
   )
```

Parâmetros

downtrend

[in] O novo valor da Propriedade "Downtrend".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_GANNFAN para CChartObjectGannFan).

© 2000-2024, MetaQuotes Ltd.

CChartObjectGannGrid

CChartObjectGannGrid é uma classe para o acesso simplificado às propriedades de objeto gráfico "Gann Grid".

Descrição

Classe CChartObjectGannGrid fornece acesso às propriedades de objeto gráfico "Gann Grid".

Declaração

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

Métodos de classe

Create	
Create	Cria o objeto gráfico "Gann Grid"
Propriedades	
PipsPerBar	Obtém/Define a propriedade "Pips per bar"
Downtrend	Obtém/Define a propriedade "Downtrend"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Gann Grid".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    ppb            // Pips per bar
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

ppb

[in] Pips por barra.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

PipsPerBar (Método Get)

Obtém o valor da propriedade "Pips per bar".

```
double  PipsPerBar() const
```

Valor de retorno

Valor da propriedade "Pips per bar" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

PipsPerBar (Método Set)

Define um novo valor à propriedade "Pips per bar".

```
bool  PipsPerBar(
   double  ppb      // Pips per bar
   )
```

Parâmetros

ppb

[in] O novo valor da propriedade "Pips per bar".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Downtrend (Método Get)

Obtém o valor da propriedade "Downtrend".

```
bool  Downtrend() const
```

Valor de retorno

Valor da propriedade do objeto "Downtrend", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Downtrend (Método Set)

Define novo valor da propriedade "Downtrend".

```
bool  Downtrend(
   bool  downtrend      // Flag value
   )
```

Parâmetros

downtrend

[in] O novo valor da Propriedade "Downtrend".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_GANNGRID para CChartObjectGannGrid).

© 2000-2024, MetaQuotes Ltd.

Ferramentas Fibonacci

Um grupo de objetos gráficos chamado "Fibonacci Tools".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Fibonacci Tools" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectFibo	Objeto gráfico "Fibonacci Retracement"
CChartObjectFiboTimes	Objeto gráfico"Fibonacci Time Zones"
CChartObjectFiboFan	Objeto gráfico "Fibonacci Fan"
CChartObjectFiboArc	Objeto gráfico "Fibonacci Arc"
CChartObjectFiboChannel	Objeto gráfico "Fibonacci Channel"
CChartObjectFiboExpansion	Objeto gráfico "Fibonacci Expansion"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectFibo

CChartObjectFibo é uma classe para o acesso simplificado às propriedades do objeto gráfico "Fibonacci".

Descrição

Classe CChartObjectFibo fornece o acesso às propriedades do objeto gráfico "Fibonacci".

Declaração

```
class CChartObjectFibo : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

Métodos de classe

Create	
Create	Cria objeto gráfico "Fibonacci Retracement"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Retracement".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2         // Second price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_FIBO para CChartObjectFibo).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboTimes

CChartObjectFiboTimes é uma classe para acesso simplificado às "propriedades do objeto gráfico "Fibonacci Time Zones".

Descrição

Classe CChartObjectFiboTimes fornece acesso às "propriedades do objeto gráfico "Fibonacci Time Zones".

Declaração

```
class CChartObjectFiboTimes : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectFiboTimes

Métodos de classe

Create	
Create	Cria o objeto gráfico "Fibonacci Time Zones"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Time Zones".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2         // Second price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_FIBOTIMES para CChartObjectFiboTimes).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboFan

CChartObjectFiboFan é uma classe para acesso simplificado às propriedades do objeto gráfico "Fibonacci Fan".

Descrição

Classe CChartObjectFiboFan fornece acesso às propriedades do objeto gráfico "Fibonacci Fan".

Declaração

```
class CChartObjectFiboFan : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectFiboFan

Métodos de classe

Create	
Create	Cria o objeto gráfico "Fibonacci Fan"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Fan".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2         // Second price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_FIBOFAN para CChartObjectFiboFan).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboArc

CChartObjectFiboArc é uma classe para acesso simplificado às propriedades do objeto gráfico "Fibonacci Arc" .

Descrição

Classe CChartObjectFiboArc fornece acesso às propriedades do objeto gráfico "Fibonacci Arc".

Declaração

```
class CChartObjectFiboArc : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectFiboArc

Métodos de classe

Create	
Create	Cria o objeto gráfico "Fibonacci Arc"
Propriedades	
Scale	Obtém/Define a propriedade "Scale"
Elipse	Obtém/Define a propriedade "Ellipse"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Arc"

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   double    scale          // Scale
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

scale

[in] Escala.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Scale (Método Get)

Obtém o valor da propriedade "Scale".

```
double  Scale() const
```

Valor de retorno

Valor da propriedade "Scale" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

Scale (Método Set)

Define um novo valor para a propriedade "Scale".

```
bool  Scale(
   double  scale      // Scale
   )
```

Parâmetros

scale

[in] O novo valor para a propriedade "Scale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Ellipse (Método Get)

Obtém o valor da propriedade "Ellipse".

```
bool  Ellipse() const
```

Valor de retorno

Valor da propriedade "Ellipse" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Ellipse (Método Set)

Define novo valor flag para a propriedade "Ellipse".

```
bool  Ellipse(
   bool  ellipse      // flag value
   )
```

Parâmetros

ellipse

[in] O novo valor para a propriedade "Scale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_FIBOARC para CChartObjectFiboArc).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboChannel

CChartObjectFiboChannel é uma classe para acesso simplificado às propriedades do objeto gráfico "Fibonacci Channel".

Descrição

Classe CChartObjectFiboChannel fornece acesso às propriedades do objeto gráfico "Fibonacci Channel".

Declaração

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

Métodos de classe

Create	
Create	Cria o objeto gráfico "Fibonacci Channel"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Channel".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3         // Third price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_FIBOCHANNEL para CChartObjectFiboChannel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectFiboExpansion

CChartObjectFiboExpansion é uma classe para acesso simplificado às propriedades do objeto gráfico"Fibonacci Expansion".

Descrição

Classe CChartObjectFiboExpansion fornece acesso às propriedades do objeto gráfico"Fibonacci Expansion".

Declaração

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Título

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

Métodos de classe

Create	
Create	Cria o objeto gráfico "Fibonacci Expansão"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Fibonacci Expansion".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3         // Third price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_EXPANSION para CChartObjectFiboExpansion).

© 2000-2024, MetaQuotes Ltd.

Ferramentas Elliott

Um grupo de objetos gráficos chamado "Elliott Tools".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Elliott Tools" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectElliottWave3	Objeto gráfico "Correcting Wave"
CChartObjectElliottWave5	Objeto gráfico"Impulse Wave"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectElliottWave3

CChartObjectElliottWave3 é uma classe para o acesso simplificado às propriedades de objetos gráficos "Correcting Wave".

Descrição

A Classe CChartObjectElliottWave3 fornece acesso às propriedades de objetos gráficos "Correcting Wave".

Declaração

```
class CChartObjectElliottWave3 : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectElliottWave3

Descendentes diretos

CChartObjectElliottWave5

Métodos de classe

Create	
Create	Cria o objeto gráfico "Correcting Wave"
Propriedades	
Degree	Obtém/Define a propriedade "Degree"
Lines	Obtém/define a propriedade "Lines"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Correcting Wave".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3         // Third price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Degree (Método Get)

Obtém o valor da propriedade "Degree".

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Valor de retorno

Valor da propriedade do objeto "Degree", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Degree (Método Set)

Define um novo valor para a propriedade "Degree".

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // property value
   )
```

Parâmetros

degree

[in] O novo valor para a propriedade "Degree".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Lines (Método Get)

Obtém o valor da propriedade "Lines".

```
bool  Lines() const
```

Valor de retorno

Valor da propriedade de objeto "Lines", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Lines (Método Set)

Define um novo valor para a propriedade "Lines".

```
bool  Lines(
   bool  lines      // flag value
   )
```

Parâmetros

lines

[in] O novo valor para a propriedade "Lines".

Valor de retorno

verdadeiro se obteve êxito, falso se a flag não mudar.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo de objeto (OBJ_ELLIOTWAVE3 para CChartObjectElliottWave3).

© 2000-2024, MetaQuotes Ltd.

CChartObjectElliottWave5

CChartObjectElliottWave5 é uma classe ao acesso simplificado às propriedades de objetos gráficos "Impulse Wave".

Descrição

Classe CChartObjectElliottWave5 fornece o acesso às propriedades de objetos gráficos "Impulse Wave".

Declaração

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Título

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

Métodos de classe

Create	
Create	Cria o objeto gráfico "Impulse Wave"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Veja também

© 2000-2024, MetaQuotes Ltd.

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Impulse Wave".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3,        // Third price coordinate
   datetime  time4,         // Fourth time coordinate
   double    price4,        // Fourth price coordinate
   datetime  time5          // Fifth time coordinate
   double    price5,        // Fifth price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

time4

[in] Coordenada de tempo do quarto ponto de ancoragem.

© 2000-2024, MetaQuotes Ltd.

price4

[in] Coordenada de preço do quarto ponto de ancoragem.

time5

[in] Coordenada de tempo do quinto ponto de ancoragem.

price5

[in] Coordenada de preço do quinto ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

identificador do tipo de objeto (OBJ_ELLIOTWAVE5 para CChartObjectElliottWave5).

© 2000-2024, MetaQuotes Ltd.

Objetos de Formas

Um grupo de objetos gráficos chamado "Shapes".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Shapes" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectRectangle	Objeto gráfico "Rectangle"
CChartObjectTriangle	Objeto gráfico "Triangle"
CChartObjectEllipse	Objeto gráfico "Ellipse"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectRectangle

CChartObjectRectangle é uma classe para acesso simplificado às propriedades do objeto gráfico "Rectangle".

Descrição

Classe CChartObjectRectangle fornece acesso às propriedades do objeto gráfico "Rectangle".

Declaração

```
class CChartObjectRectangle : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectRectangle

Métodos de classe

Create	
Create	Cria o objeto gráfico "Rectangle"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Rectangle".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2         // Second price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

Valor de retorno

verdadeiro se bem-sucedido, 0 se houver erro

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_RECTANGLE para CChartObjectRectangle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectTriangle

CChartObjectTriangle é uma classe de acesso simplificado às propriedades do objeto gráfico para "Triangle".

Descrição

Classe CChartObjectTriangle fornece acesso às propriedades do objeto gráfico para "Triangle".

Declaração

```
class CChartObjectTriangle : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectTriangle

Métodos de classe

Create	
Create	Cria o objeto gráfico "Triangle"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Triangle".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   long      window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3         // Third price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se ocorrer erro

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_TRIANGLE para CChartObjectTriangle).

© 2000-2024, MetaQuotes Ltd.

CChartObjectEllipse

Classe CChartObjectEllipse é uma classe de acesso simplificado às propriedades do objeto gráfico "Ellipse".

Descrição

Classe CChartObjectEllipse fornece acesso às propriedades do objeto gráfico "Ellipse".

Declaração

```
class CChartObjectEllipse : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectEllipse

Métodos de classe

Create	
Create	Cria o objeto gráfico "Ellipse"
Entrada/saída	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Ellipse".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time1,         // First time coordinate
   double    price1,        // First price coordinate
   datetime  time2,         // Second time coordinate
   double    price2,        // Second price coordinate
   datetime  time3,         // Third time coordinate
   double    price3         // Third price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time1

[in] Coordenada de tempo do primeiro ponto de ancoragem.

price1

[in] Coordenada de preço do primeiro ponto de ancoragem.

time2

[in] Coordenada de tempo do segundo ponto de ancoragem.

price2

[in] Coordenada de preço do segundo ponto de ancoragem.

time3

[in] Coordenada de tempo do terceiro ponto de ancoragem.

price3

[in] Coordenada de preço do terceiro ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se ocorrer erro

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_ELLIPSE para CChartObjectEllipse).

© 2000-2024, MetaQuotes Ltd.

Objetos de Setas

Grupo para objetos gráficos chamado "Arrows".

Esta seção contém os detalhes técnicos de trabalho com um grupo de classes de objetos gráficos chamado "Arrow" e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5. Em essência, o "arrow" é um ícone a ser exibido ao usuário, correspondendo a um determinado código. Existem dois tipos de objetos gráficos "Arrow" para exibir ícones nos gráficos:

- Objeto "Arrow" que permite especificar o código do ícone exibido no gráfico.
- Objetos de grupo para exibir certos tipos de ícones (e o preciso código fixo correspondente).

Classe para "arrow" exibe ícones de código arbitrário

Nome da classe	Nome do objeto arrow
CChartObjectArrow	Arrow

Classe do código fixo para o ícone "arrow"

Nome da classe	Nome do objeto arrow
CChartObjectArrowCheck	Verificar
CChartObjectArrowDown	Seta para cima
CChartObjectArrowUp	Seta para baixo
CChartObjectArrowStop	Sinal de Parada
CChartObjectArrowThumbDown	Polegar para cima
CChartObjectArrowThumbUp	Polegar para Baixo
CChartObjectArrowLeftPrice	Etiqueta de Preço à Esquerda
CChartObjectArrowRightPrice	Etiqueta de Preço Direito

Veja também

Objetos tipo, Métodos de Vinculação de Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

CChartObjectArrow

Classe CChartObjectArrow é uma classe para acesso simplificado a propriedades de objetos gráficos "Arrow".

Descrição

Classe CChartObjectArrow fornece o acesso a propriedades comuns de objetos "Arrow" para todos os seus descendentes.

Declaração

```
class CChartObjectArrow : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectArrow

Descendentes diretos

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

Métodos de classe

Create	
Create	Cria objeto gráfico "Arrow"
Propriedades	
ArrowCode	Obtém/Define propriedade "Arrow Code"
Anchor	Obtém/Define propriedade "Anchor"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Objetos tipo, Métodos de Vinculação de Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria objeto de gráfico "Arrow".

```
bool  Create(
   long      chart_id,      // Chart ID
   string    name,          // Object Name
   int       window,        // Chart Window
   datetime  time,          // Time
   double    price,         // Price
   char      code           // Arrow Code
   )
```

Parâmetros

chart_id

[in] Identificador de Gráfico (0 - gráfico corrente).

name

[in] Nome de objeto (deve ser único).

window

[in] Número de janela do gráfico (0 - janela de base).

time

[in] Coordenada de tempo.

price

[in] Coordenada de preço

code

[in] Código "Arrow" (Wingdings).

Valor de retorno

verdadeiro - se bem sucedido, caso contrário falso.

Exemplo

```
//--- example for CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- arrow create error
      printf("Arrow create: Error %d!",GetLastError());
      //---
```

© 2000-2024, MetaQuotes Ltd.

```
      return;
     }
//--- use arrow
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrowCode (Método Get)

Obtém o código do símbolo para "Arrow".

```
char  ArrowCode() const
```

Valor de retorno

Código de símbolo do objeto "Arrow", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

ArrowCode (Método Set)

Define código do símbolo "Arrow"

```
bool  ArrowCode(
   char  code      // Code value
   )
```

Parâmetros

code

[in] novo valor para o código "arrow" (Wingdings).

Valor de retorno

verdadeiro - se bem sucedido, falso - se o código não foi alterado.

Exemplo

```
//--- example for CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- arrow create error
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- use arrow to possible changes code
//--- . . .
//--- get code of arrow
   if(arrow.ArrowCode()!=code)
     {
     //--- set code of arrow
```

© 2000-2024, MetaQuotes Ltd.

```
      arrow.ArrowCode(code);
      }
//--- use arrow
//--- . . .
   }
```

© 2000-2024, MetaQuotes Ltd.

Anchor (Get Method)

Obtém o tipo de âncora do objeto "Arrow"

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Valor de retorno

Tipo de âncora do objeto "Arrow", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Anchor (Set Method)

Define o tipo de âncora para o objeto "Arrow"

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // new anchor type
   )
```

Parâmetros

anchor

[in] Novo tipo de âncora

Valor de retorno

verdadeiro se obteve êxito, falso se o tipo de âncora não mudou.

Exemplo

```
//--- example for CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- arrow create error
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- get anchor of arrow
   if(arrow.Anchor()!=anchor)
     {
     //--- set anchor of arrow
     arrow.Anchor(anchor);
     }
```

© 2000-2024, MetaQuotes Ltd.

```
//--- use arrow
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // file handle
   )
```

Parâmetros

file_handle

[in] Manipula o arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Exemplo

```
//--- example for CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- arrow create error
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- open file
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- file save error
         printf("File save: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // file handle
   )
```

Parâmetros

file_handle

[in] manipula o arquivo já aberto usando a função FileOpen (...).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Exemplo

```
//--- example for CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- open file
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- file load error
         printf("File load: Error %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- use arrow
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (por exemplo, para OBJ_ARROW CChartObjectArrow)

Exemplo

```
//--- example for CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- get arrow type
   int type=arrow.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Setas com código fixo

Classes "Setas com código fixo" servem para o acesso simplificado às propriedades dos seguintes objetos gráficos:

Nome da classe	Nome do objeta tipo Seta
CChartObjectArrowCheck	"Arrow Check" (Verificar Seta)
CChartObjectArrowDown	"Arrow Down" (Seta Abaixo)
CChartObjectArrowUp	"Arrow Up" (Seta Acima)
CChartObjectArrowStop	"Arrow Stop" (Seta Parar)
CChartObjectArrowThumbDown	"Good" ("Big finger up") - Sinal Positivo com Dedão Acima
CChartObjectArrowThumbUp	"Bad" ("Big finger down")- Sinal Negativo com Dedão Abaixo
CChartObjectArrowLeftPrice	"Left price" (preço a esquerda)
CChartObjectArrowRightPrice	"Right price" (preço a direita)

Descrição

Classe "Setas com código fixo" fornece acesso às propriedades do objeto.

Declarações

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Título

```
<ChartObjects\ChartObjectsArrows.mqh>
```

Métodos de classe

Create	
Create	Cria um objeto gráfico específico
Propriedades	
ArrowCode	"Stub" para o método de mudança do código

© 2000-2024, MetaQuotes Ltd.

Create	
Entrada/saída	
virtual Type	Método virtual de identificação

Veja também

Objetos tipo, Métodos de Vinculação de Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria objeto gráfico tipo "Seta com código fixo".

```
bool  Create(
   long      chart_id,      // Chart ID
   string    name,          // Object Name
   int       window,        // Chart Window
   datetime  time,          // Time
   double    price          // Price
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Único nome do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time

[in] Coordenada de tempo.

price

[in] Coordenada de preço

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

Exemplo

```
//--- example for CChartObjectArrowCheck::Create
//--- example for CChartObjectArrowDown::Create
//--- example for CChartObjectArrowUp::Create
//--- example for CChartObjectArrowStop::Create
//--- example for CChartObjectArrowThumbDown::Create
//--- example for CChartObjectArrowThumbUp::Create
//--- example for CChartObjectArrowLeftPrice::Create
//--- example for CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- for example, take CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2024, MetaQuotes Ltd.

```
      {
       //--- arrow create error
       printf("Arrow create: Error %d!",GetLastError());
       //---
       return;
      }
//--- use arrow
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

ArrowCode

Proíbe alterações de código tipo "Arrow".

```
bool  ArrowCode(
   char  code      // code value
   )
```

Parâmetros

code

[in] qualquer valor

Valor de retorno

Sempre falso.

Exemplo

```
//--- example for CChartObjectArrowCheck::ArrowCode
//--- example for CChartObjectArrowDown::ArrowCode
//--- example for CChartObjectArrowUp::ArrowCode
//--- example for CChartObjectArrowStop::ArrowCode
//--- example for CChartObjectArrowThumbDown::ArrowCode
//--- example for CChartObjectArrowThumbUp::ArrowCode
//--- example for CChartObjectArrowLeftPrice::ArrowCode
//--- example for CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- for example, take CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- set object parameters
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- arrow create error
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- set code of arrow
   if(!arrow.ArrowCode(181))
     {
      //--- it is not error
      printf("Arrow code can not be changed");
     }
//--- use arrow
//--- . . .
  }
```

© 2000-2024, MetaQuotes Ltd.

Type

Retorna gráfico tipo identificador de objeto

```
virtual int  Type() const
```

Valor de retorno

Tipo identificador de objeto(OBJ_ARROW_CHECK for CChartObjectArrowCheck, OBJ_ARROW_DOWN for CChartObjectArrowDown, OBJ_ARROW_UP for CChartObjectArrowUp, OBJ_ARROW_STOP for CChartObjectArrowStop, OBJ_ARROW_THUMB_DOWN for CChartObjectArrowThumbDown, OBJ_ARROW_THUMB_UP for CChartObjectArrowThumbUp, OBJ_ARROW_LEFT_PRICE for CChartObjectArrowLeftPrice, OBJ_ARROW_RIGHT_PRICE for CChartObjectArrowRightPrice).

Exemplo

```
//--- example for CChartObjectArrowCheck::Type
//--- example for CChartObjectArrowDown::Type
//--- example for CChartObjectArrowUp::Type
//--- example for CChartObjectArrowStop::Type
//--- example for CChartObjectArrowThumbDown::Type
//--- example for CChartObjectArrowThumbUp::Type
//--- example for CChartObjectArrowLeftPrice::Type
//--- example for CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- for example, take CChartObjectArrowCheck
   CChartObjectArrowCheck arrow;
//--- get arrow type
   int type=arrow.Type();
  }
```

© 2000-2024, MetaQuotes Ltd.

Objetos de Controles

Um grupo de objetos gráficos chamado "Object Controls".

Esta seção contém os detalhes técnicos de trabalhar com um grupo de objetos gráficos chamado "Object Controls e uma descrição dos componentes relevantes da Biblioteca Padrão MQL5.

Nome da classe	Objeto
CChartObjectText	Objeto gráfico "Text"
CChartObjectLabel	Objeto gráfico "Text Label"
CChartObjectEdit	Objeto gráfico "Edit"
CChartObjectButton	Objeto gráfico "Button"
CChartObjectSubChart	Objeto gráfico "Chart"
CChartObjectBitmap	Objeto gráfico "Bitmap"
CChartObjectBmpLabel	Objeto gráfico "Bitmap Label"
CChartObjectRectLabel	Objeto gráfico "Rectangle Label"

Veja também

Tipos de objeto, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

CChartObjectText

CChartObjectText é uma classe de acesso simplificado às propriedade do objeto gráfico "Text".

Descrição

Classe CChartObjectText fornece acesso às propriedade do objeto gráfico "Text".

Declaração

```
class CChartObjectText : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectText

Descendentes diretos

CChartObjectLabel

Métodos de classe

Create	
Create	Cria objeto gráfico "Text"
Propriedades	
Angle	Obtém/Define a propriedade "Angle"
Font	Obtém/Define a propriedade "Font"
FontSize	Obtém/Define a propriedade "FontSize"
Anchor	Obtém/Define propriedade "Anchor"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

As classes derivadas:

- CChartObjectLabel

Veja também

Tipos de objetos, Propriedades de objetos, Métodos de Vinculação de Objeto, Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Text".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time,          // Time coordinate
   double    price          // Price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time

[in] Coordenadas de tempo do ponto de ancoragem.

price

[in] Coordenadas de preço do ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Angle (Método Get)

Obtém o valor da propriedade "Angle".

```
double  Angle() const
```

Valor de retorno

Valor da propriedade "Angle" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, retorna EMPTY_VALUE.

Angle (Método Set)

Define novo valor da propriedade "Angle".

```
bool  Angle(
   double  angle      // new angle
   )
```

Parâmetros

angle

[in] O novo valor da propriedade "Angle".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Font (Método Get)

Obtém o valor da propriedade "Font".

```
string  Font() const
```

Valor de retorno

Valor da propriedade "Font" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna "".

Font (Método Set)

Define novo valor da propriedade "Font".

```
bool  Font(
   string  font      // new font
   )
```

Parâmetros

font

[in] O novo valor da propriedade "Font".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

FontSize (Método Get)

Obtém o valor da propriedade "FontSize".

```
int  FontSize() const
```

Valor de retorno

Valor da propriedade "FontSize" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

FontSize (Método Set)

Define novo valor da propriedade "FontSize".

```
bool  FontSize(
   int  size      // new font size
   )
```

Parâmetros

size

[in] O novo valor da propriedade "Font".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Anchor (Método Get)

Obtém o valor da propriedade "Anchor".

```
ENUM_ANCHOR_POINT  Anchor() const
```

Valor de retorno

Valor da propriedade "Anchor" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Anchor (Método Set)

Define novo valor da propriedade "Anchor".

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // new value
   )
```

Parâmetros

anchor

[in] O novo valor da propriedade "Anchor".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_TEXT para CChartObjectText).

© 2000-2024, MetaQuotes Ltd.

CChartObjectLabel

CChartObjectLabel é uma classe de acesso simplificado às propriedades do objeto gráfico "Label" .

Descrição

Classe CChartObjectLabel fornece acesso às propriedades do objeto gráfico "Label"

Declaração

```
class CChartObjectLabel : public CChartObjectText
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

Descendentes diretos

CChartObjectEdit, CChartObjectRectLabel

Métodos de classe

Create	
Create	Cria o objeto gráfico "Label"
Propriedades	
X_Distance	Obtém/Define a propriedade "X_Distance"
Y_Distance	Obtém/Define a propriedade "Y_Distance"
X_Size	Obtém/Define a propriedade "X_Size"
Y_Size	Obtém/Define propriedade "Y_Size"
Corner	Obtém/Define a propriedade "Corner"
Time	"Stub" para mudar a Coordenada de Tempo
Price	"Stub" para mudar a Coordenada de Preço
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Veja também

Tipos de objetos, Propriedades de objetos, Ângulo de gráfico, Métodos de Vinculação de Objeto, Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Label".

```
bool  Create(
   long    chart_id,      // Chart identifier
   string  name,          // Object name
   int     window,        // Chart window
   int     X,             // X coordinate
   int     Y              // Y coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Método Get)

Obtém o valor da propriedade "X_Distance".

```
int  X_Distance() const
```

Valor de retorno

Valor da propriedade "X_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Distance (Método Set)

Define novo valor da propriedade "X_Distance".

```
bool  X_Distance(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtém o valor da propriedade "Y_Distance".

```
int  Y_Distance() const
```

Valor de retorno

Valor da propriedade "Y_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Distance (Método Set)

Define novo valor da propriedade "Y_Distance".

```
bool  Y_Distance(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor da propriedade "Y_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Size

Obtém o valor da propriedade "X_Size".

```
int  X_Size() const
```

Valor de retorno

Valor da propriedade "X_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Obtém o valor da propriedade "Y_Size".

```
int  Y_Size() const
```

Valor de retorno

Valor da propriedade "Y_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

Corner (Método Get)

Obtém o valor da propriedade "Corner".

```
ENUM_BASE_CORNER  Corner() const
```

Valor de retorno

Valor da propriedade "Corner" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Corner (Método Set)

Define novo valor da propriedade "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // new value
   )
```

Parâmetros

corner

[in] O novo valor da propriedade "Corner".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Time

Proíbe mudanças da coordenada do tempo.

```
bool  Time(
   datetime  time      // any value
   )
```

Parâmetros

time

[in] Qualquer valor do tipo "datetime".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Price

Proíbe mudanças da coordenada do preço.

```
bool  Price(
   double  price      // any value
   )
```

Parâmetros

price

[in] Qualquer valor do tipo "double".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_LABEL para CChartObjectLabel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectEdit

CChartObjectEdit é uma classe de acesso simplificado às propriedades do objeto gráfico "Edit".

Descrição

Classe CChartObjectEdit fornece acesso às propriedades do objeto gráfico"Edit".

Declaração

```
class CChartObjectEdit : public CChartObjectLabel
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit

Descendentes diretos

CChartObjectButton

Métodos de classe

Create	
Create	Cria o objeto gráfico "Edit"
Propriedades	
TextAlign	Obtém/Define a propriedade "TextAlign"
X_Size	Obtém a propriedade "X Size"
Y_Size	Obtém a propriedade "Y Size"
BackColor	Obtém/define a propriedade "Background Color"
BorderColor	Obtém/define a propriedade "Border Color"
ReadOnly	Obtém/define a propriedade "Read Only"
Angle	Obtém/define a propriedade "Angle"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo

© 2000-2024, MetaQuotes Ltd.

Create	
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos herdados da classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Veja também

Tipos de objetos, Propriedades de objetos, Ângulo de gráfico, Métodos de Vinculação de Objeto, Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Edit".

```
bool  Create(
   long    chart_id,      // Chart identifier
   string  name,          // Object name
   int     window,        // Chart window
   int     X,             // X coordinate
   int     Y,             // Y coordinate
   int     sizeX,         // X size
   int     sizeY          // Y size
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamanho de X.

sizeY

[in] Tamanho de Y.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

TextAlign (Método Get)

Obtém o valor da propriedade "TextAlign" (modo de alinhamento do texto).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valor de retorno

Valor da propriedade "TextAlign" do objeto, atribuído à instância de classe.

TextAlign (Método Set)

Define o valor da propriedade "TextAlign" (Alinhamento modo texto).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // new value
   )
```

Parâmetros

align

[in] O novo valor da propriedade "TextAlign".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Size

Define novo valor da propriedade "X_Size".

```
bool  X_Size(
   int  size      // new size
   )
```

Parâmetros

size

[in] O novo valor da propriedade "X_Size".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Define novo valor da propriedade "Y_Size".

```
bool  Y_Size(
   int  size      // new size
   )
```

Parâmetros

size

[in] O novo valor da propriedade "Y_Size".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

BackColor (Método Get)

Obtém o valor da propriedade "BackColor".

```
color  BackColor() const
```

Valor de retorno

Valor da propriedade "BackColor" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna CLR_NONE.

BackColor (Método Set)

Define novo valor da propriedade "BackColor".

```
bool  BackColor(
   color  new_color      // new background color
   )
```

Parâmetros

new_color

[in] O novo valor da propriedade "BackColor".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

BorderColor (Método Get)

Obtém o valor da propriedade "Border Color".

```
color  BorderColor() const
```

Valor de retorno

Valor da propriedade "Border Color" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna CLR_NONE.

BorderColor (Método Set)

Define novo valor da propriedade "Border Color".

```
bool  BorderColor(
   color  new_color      // new border color
   )
```

Parâmetros

new_color

[in] O novo valor da propriedade "Border Color".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ReadOnly (Método Get)

Obtém o valor da propriedade "Read Only".

```
bool  ReadOnly() const
```

Valor de retorno

O valor da propriedade do objeto "Read Only", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

ReadOnly (Método Set)

Define novo valor da propriedade "Read Only.

```
bool  ReadOnly(
   const  bool  flag      // new value
   )
```

Parâmetros

flag

[in] O novo valor para da propriedade "Read Only.

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Angle

Proíbe alterações da propriedade "Angle".

```
bool  Angle(
   double  angle      // any value
   )
```

Parâmetros

angle

[in] Qualquer valor do tipo "double".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_EDIT para CChartObjectEdit).

© 2000-2024, MetaQuotes Ltd.

CChartObjectButton

CChartObjectButton é uma classe simplificada de acesso às propriedades do objeto gráfico "Button".

Descrição

Class CChartObjectButton fornece acesso às propriedades do objeto gráfico "Button".

Declaração

```
class CChartObjectButton : public CChartObjectEdit
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectEdit
          CChartObjectButton

Descendentes diretos

CChartObjectPanel

Métodos de classe

Create	
Create	Formulário herdado da CChartObjectEdit
Propriedades	
State	Obtém/Define estado do botão (Ligado/Desligado)
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos herdados da classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Métodos herdados da classe CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Veja também

Tipos de objetos, Propriedades de objetos, Ângulo de gráfico, Métodos de Vinculação de Objeto, Objetos Gráficos

© 2000-2024, MetaQuotes Ltd.

State (Método Get)

Obtém o valor da propriedade "State".

```
bool  State() const
```

Valor de retorno

Valor da propriedade "State" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

State (Método Set)

Define novo valor da propriedade "State".

```
bool  State(
   bool  state      // new state value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "State".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_BUTTON para CChartObjectButton).

© 2000-2024, MetaQuotes Ltd.

CChartObjectSubChart

CChartObjectSubChart é uma classe de acesso simplificado às propriedades do objeto gráfico "Chart".

Descrição

Classe CChartObjectSubChart fornece acesso às propriedades do objeto gráfico "Chart".

Declaração

```
class CChartObjectSubChart : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectSubChart

Métodos de classe

Create	
Create	Cria o objeto gráfico "Chart"
Propriedades	
X_Distance	Obtém/Define a propriedade "X_Distance"
Y_Distance	Obtém/Define a propriedade "Y_Distance"
Corner	Obtém/Define a propriedade "Corner"
X_Size	Obtém/Define a propriedade "X_Size"
Y_Size	Obtém/Define propriedade "Y_Size"
Symbol	Obtém/Define a propriedade "Symbol"
Period	Obtém/Define a propriedade "Period"
Scale	Obtém/Define a propriedade "Scale"
DateScale	Obtém/Define propriedade "Show date scale"
PriceScale	Obtém/Define propriedade "Show price scale"
Time	"Stub" para alteração da coordenada de tempo
Price	"Stub" para alteração da coordenada de preço
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo

© 2000-2024, MetaQuotes Ltd.

Create	
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objetos, Propriedades de objetos, ângulo de gráfico, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "SubChart".

```
bool  Create(
   long    chart_id,      // Chart identifier
   string  name,          // Object name
   int     window,        // Chart window
   int     X,             // X coordinate
   int     Y,             // Y coordinate
   int     sizeX,         // X size
   int     sizeY          // Y size
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamanho de X.

sizeY

[in] Tamanho de Y.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Método Get)

Obtém o valor da propriedade "X_Distance".

```
int  X_Distance() const
```

Valor de retorno

Valor da propriedade "X_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Distance (Método Set)

Define novo valor da propriedade "X_Distance".

```
bool  X_Distance(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtém o valor da propriedade "Y_Distance".

```
int  Y_Distance() const
```

Valor de retorno

Valor da propriedade "Y_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Distance (Método Set)

Define novo valor da propriedade "Y_Distance".

```
bool  Y_Distance(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor da propriedade "Y_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Corner (Método Get)

Obtém o valor da propriedade "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valor de retorno

Valor da propriedade "Corner" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Corner (Método Set)

Define novo valor da propriedade "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // new value
   )
```

Parâmetros

corner

[in] O novo valor da propriedade "Corner".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Size (Método Get)

Obtém o valor da propriedade "X_Size".

```
int  X_Size() const
```

Valor de retorno

Valor da propriedade "X_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Size (Método Set)

Define novo valor da propriedade "X_Size".

```
bool  X_Size(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Size".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Size (Método Get)

Obtém o valor da propriedade "Y_Size".

```
int  Y_Size() const
```

Valor de retorno

Valor da propriedade "Y_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Size (Método Set)

Define novo valor da propriedade "Y_Size".

```
bool  Y_Size(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor da propriedade "Y_Size".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Symbol (Método Get)

Obtém o valor da propriedade "Symbol".

```
string  Symbol() const
```

Valor de retorno

Valor da propriedade "Symbol" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna "".

Symbol (Método Set)

Define novo valor da propriedade "Symbol".

```
bool  Symbol(
   string  symbol      // new symbol
   )
```

Parâmetros

symbol

[in] O novo valor da propriedade "Símbolo".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Period (Método Get)

Obtém o valor da propriedade "Period".

```
int  Period() const
```

Valor de retorno

Valor da propriedade "Period" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Period (Método Set)

Define novo valor da propriedade "Period".

```
bool  Period(
   int  period      // new period
   )
```

Parâmetros

period

[in] O novo valor da propriedade "Period".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Scale (Método Get)

Obtém o valor da propriedade "Scale".

```
double  Scale() const
```

Valor de retorno

Valor da propriedade do objeto "Scale", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna EMPTY_VALUE.

Scale (Método Set)

Define um novo valor para a propriedade "Scale".

```
bool  Scale(
   double  scale      // new scale
   )
```

Parâmetros

scale

[in] O novo valor para a propriedade "Scale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

DateScale (Método Get)

Obtém o valor da propriedade "DateScale".

```
bool  DateScale() const
```

Valor de retorno

Valor da Propriedade do objeto "DateScale", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

DateScale (Método Set)

Define novo valor da propriedade "DateScale".

```
bool  DateScale(
   bool  scale      // new value
   )
```

Parâmetros

scale

[in] O novo valor da propriedade "DateScale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

PriceScale (Método Get)

Obtém o valor da propriedade "PriceScale".

```
bool  PriceScale() const
```

Valor de retorno

Valor da propriedade do objeto "PriceScale", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

PriceScale (Método Set)

Define novo valor da propriedade "PriceScale".

```
bool  PriceScale(
   bool  scale      // new value
   )
```

Parâmetros

scale

[in] O novo valor da propriedade "PriceScale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Time

Proíbe mudanças da coordenada do tempo.

```
bool  Time(
   datetime  time      // any value
   )
```

Parâmetros

time

[in] Qualquer valor do tipo "datetime".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Price

Proíbe mudanças da coordenada do preço.

```
bool  Price(
   double  price      // any value
   )
```

Parâmetros

price

[in] Qualquer valor do tipo "double".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_CHART para CChartObjectSubChart).

© 2000-2024, MetaQuotes Ltd.

CChartObjectBitmap

CChartObjectBitmap é uma classe para acesso simplificado às propriedades do objeto gráfico "Bitmap".

Descrição

Classe CChartObjectBitmap fornece acesso às propriedades do objeto gráfico "Bitmap".<

Declaração

```
class CChartObjectBitmap : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectBitmap

Métodos de classe

Create	
Create	Cria o objeto gráfico "Bitmap"
Propriedades	
BmpFile	Obtém/Define a propriedade "BMP Filename"
X_Offset	Obtém/Define a propriedade "X_Offset"
Y_Offset	Obtém/Define a propriedade "X_Offset"
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objetos, Propriedades de objetos, Objetos gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "Bitmap".

```
bool  Create(
   long      chart_id,      // Chart identifier
   string    name,          // Object name
   int       window,        // Chart window
   datetime  time,          // Time coordinate
   double    price          // Price coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto para criar.

window

[in] Número de janela do gráfico (0 - janela de base).

time

[in] Coordenadas de tempo do ponto de ancoragem.

price

[in] Coordenadas de preço do ponto de ancoragem.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

BmpFile (Get Method)

Obtém o valor da propriedade "BmpFile".

```
string  BmpFile() const
```

Valor de retorno

Valor da propriedade "BmpFile" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

BmpFile (Set Method)

Define um novo valor à propriedade "BmpFile".

```
bool  BmpFile(
   string  name      // new file name
   )
```

Parâmetros

name

[in] O novo valor da propriedade "BmpFile".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Offset (Método Get)

Obtém o valor da propriedade "X_Offset" (coordenada X do canto superior esquerdo da área retangular visível nos objetos gráficos).

```
int  X_Offset() const
```

Valor de retorno

Valor da propriedade "X_Offset" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Offset (Método Set)

Define um novo valor para a propriedade "X_Offset" (coordenada X do canto superior esquerdo da área retangular visível nos objetos gráficos). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.

```
bool  X_Offset(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Offset".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Offset (Método Get)

Obtém o valor da propriedade "Y_Offset" (coordenada Y do canto superior esquerdo da área retangular visível nos objetos gráficos).

```
int  Y_Offset() const
```

Valor de retorno

Valor da propriedade "Y_Offset" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Offset (Método Set)

Define um novo valor para a propriedade "Y_Offset" (coordenada Y do canto superior esquerdo da área retangular visível nos objetos gráficos). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.

```
bool  Y_Offset(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor para a propriedade "Y_Offset".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto(OBJ_BITMAP para CChartObjectBitmap).

© 2000-2024, MetaQuotes Ltd.

CChartObjectBmpLabel

CChartObjectBmpLabel é uma classe para acesso simplificado às propriedades do objeto gráfico "Bitmap Label".

Descrição

Classe CChartObjectBmpLabel fornece acesso às propriedades do objeto gráfico "Bitmap Label".

Declaração

```
class CChartObjectBmpLabel : public CChartObject
```

Título

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectBmpLabel

Métodos de classe

Create	
Create	Cria o objeto gráfico "BmpLabel"
Propriedades	
X_Distance	Obtém/Define a propriedade "X_Distance"
Y_Distance	Obtém/Define a propriedade "Y_Distance"
X_Offset	Obtém/Define a propriedade "X_Offset"
Y_Offset	Obtém/Define a propriedade "X_Offset"
Corner	Obtém/Define a propriedade "Corner"
X_Size	Obtém/Define a propriedade "X_Size"
Y_Size	Obtém/Define propriedade "Y_Size"
BmpFileOn	Obtém/Define a propriedade "BmpFileOn" quando o botão estiver ligado (On)
BmpFileOff	Obtém/Define a propriedade "BmpFileOff" quando o botão estiver desligado (Off)
State	Obtém/Define a propriedade "Button State" (Ligado/Desligado)
Time	"Stub" para alteração da coordenada de tempo
Price	"Stub" para alteração da coordenada de preço

© 2000-2024, MetaQuotes Ltd.

Create	
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Veja também

Tipos de objetos, Propriedades de objetos, ângulo de gráfico, Objetos gráfico

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "BmpLabel".

```
bool  Create(
   long    chart_id,      // Chart identifier
   string  name,          // Object name
   int     window,        // Chart window
   int     X,             // X coordinate
   int     Y              // Y coordinate
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto a criar.

window

[in] Número de janela do gráfico (0 - janela de base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

X_Distance (Método Get)

Obtém o valor da propriedade "X_Distance".

```
int  X_Distance() const
```

Valor de retorno

Valor da propriedade "X_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Distance (Método Set)

Define novo valor da propriedade "X_Distance".

```
bool  X_Distance(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Distance (Método Get)

Obtém o valor da propriedade "Y_Distance".

```
int  Y_Distance() const
```

Valor de retorno

Valor da propriedade "Y_Distance" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Distance (Método Set)

Define novo valor da propriedade "Y_Distance".

```
bool  Y_Distance(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor da propriedade "Y_Distance".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Offset (Método Get)

Obtém o valor da propriedade "X_Offset" (coordenada X do canto superior esquerdo da área retangular visível nos objetos gráficos).

```
int  X_Offset() const
```

Valor de retorno

Valor da propriedade "X_Offset" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

X_Offset (Método Set)

Define um novo valor para a propriedade "X_Offset" (coordenada X do canto superior esquerdo da área retangular visível nos objetos gráficos). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.

```
bool  X_Offset(
   int  X      // new value
   )
```

Parâmetros

X

[in] O novo valor da propriedade "X_Offset".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Y_Offset (Método Get)

Obtém o valor da propriedade "Y_Offset" (coordenada Y do canto superior esquerdo da área retangular visível nos objetos gráficos).

```
int  Y_Offset() const
```

Valor de retorno

Valor da propriedade "Y_Offset" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Y_Offset (Método Set)

Define um novo valor para a propriedade "Y_Offset" (coordenada Y do canto superior esquerdo da área retangular visível nos objetos gráficos). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.

```
bool  Y_Offset(
   int  Y      // new value
   )
```

Parâmetros

Y

[in] O novo valor para a propriedade "Y_Offset".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Corner (Método Get)

Obtém o valor da propriedade "Corner".

```
ENUM_BASE_CORNER Corner() const
```

Valor de retorno

Valor da propriedade "Corner" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Corner (Set Method)

Define novo valor da propriedade "Corner".

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // new value
   )
```

Parâmetros

corner

[in] O novo valor da propriedade "Corner".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

X_Size

Obtém o valor da propriedade "X_Size".

```
int  X_Size() const
```

Valor de retorno

Valor da propriedade "X_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Obtém o valor da propriedade "Y_Size".

```
int  Y_Size() const
```

Valor de retorno

Valor da propriedade "Y_Size" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

BmpFileOn (Método Get)

Obtém o valor da propriedade "BmpFileOn".

```
string  BmpFileOn() const
```

Valor de retorno

Valor da propriedade "BmpFileOn" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna "".

BmpFileOn (Método Set)

Define novo valor da propriedade "BmpFileOn".

```
bool  BmpFileOn(
   string  name      // file name
   )
```

Parâmetros

name

[in] O novo valor da propriedade "BmpFileOn".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

BmpFileOff (Método Get)

Obtém o valor da propriedade "BmpFileOff".

```
string  BmpFileOff() const
```

Valor de retorno

Valor da propriedade "BmpFileOff" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna "".

BmpFileOff (Método Set)

Define novo valor da propriedade "BmpFileOff".

```
bool  BmpFileOff(
   string  name      // file name
   )
```

Parâmetros

name

[in] O novo valor da propriedade "BmpFileOff".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

State (Get Method)

Obtém o valor da propriedade "State".

```
bool  State() const
```

Valor de retorno

Valor da propriedade "State" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

State (Set Method)

Define novo valor da propriedade "State".

```
bool  State(
   bool  state      // new state value
   )
```

Parâmetros

state

[in] O novo valor da propriedade "State".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Time

Proíbe mudanças da coordenada do tempo.

```
bool  Time(
   datetime  time      // any value
   )
```

Parâmetros

time

[in] Qualquer valor do tipo "datetime".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Price

Proíbe mudanças da coordenada do preço.

```
bool  Price(
   double  price      // any value
   )
```

Parâmetros

price

[in] Qualquer valor do tipo "double".

Valor de retorno

sempre falso.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_BITMAP_LABEL para CChartObjectBmpLabel).

© 2000-2024, MetaQuotes Ltd.

CChartObjectRectLabel

CChartObjectRectLabel é uma classe de acesso simplificado às propriedades do objeto gráfico "Rectangle Label".

Descrição

Classe CChartObjectRectLabel fornece acesso às propriedades do objeto gráfico "Rectangle Label".

Declaração

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Título

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Hierarquia de herança

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

Métodos de classe

Create	
Create	Cria o objeto gráfico "RectLabel"
Propriedades	
X_Size	Define o tamanho horizontal
Y_Size	Define o tamanho vertical
BackColor	Obtém/Define a cor de segundo plano (fundo)
Angle	Um método "Stub"
BorderType	Obtém/Define o tipo de borda
Entrada/saída	
virtual Save	Método virtual para escrever o arquivo
virtual Load	Método virtual para a leitura a partir do arquivo
virtual Type	Método virtual de identificação

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CChartObject

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Métodos herdados da classe CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Métodos herdados da classe CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Veja também

Tipos de objetos, Propriedades de objetos, Objetos gráficos

© 2000-2024, MetaQuotes Ltd.

Create

Cria o objeto gráfico "CChartObjectRectLabel".

```
bool  Create(
   long    chart_id,      // ID Grafico
   string  name,          // Object name
   int     window,        // Chart window
   int     X,             // X coordinate
   int     Y,             // Y coordinate
   int     sizeX,         // Horizontal size
   int     sizeY          // Vertical size
   )
```

Parâmetros

chart_id

[in] Identificador de gráfico (0 - gráfico corrente).

name

[in] Um nome exclusivo do objeto a criar.

window

[in] Número de janela do gráfico (0 - janela de base).

X

[in] Coordenada X.

Y

[in] Coordenada Y.

sizeX

[in] Tamanho horizontal.

sizeY

[in] Tamanho vertical.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

X_Size

Define o valor da propriedade "X_Size".

```
bool  X_Size(
   int  size      // Horizontal size
   )
```

Parâmetros

size

[in] Novo tamanho horizontal.

Valor de retorno

veradeiro se obteve êxito, falso se a propriedade não foi alterada.

Observação

Para obter os valores das propriedades X_Size e Y_Size usar o método de classe pai CChartObjectLabel.

© 2000-2024, MetaQuotes Ltd.

Y_Size

Define o valor da propriedade "Y_Size".

```
bool  Y_Size(
   int  size      // Vertical size
   )
```

Parâmetros

size

[in] Novo tamanho vertical.

Valor de retorno

veradeiro se obteve êxito, falso se a propriedade não foi alterada.

Observação

Para obter os valores das propriedades X_Size e Y_Size usar o método de classe pai CChartObjectLabel.

© 2000-2024, MetaQuotes Ltd.

BackColor

Obtém a cor de segundo plano (fundo).

```
color  BackColor() const
```

Valor de retorno

A cor do fundo do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

BackColor

Define a cor de segundo plano (fundo).

```
bool  BackColor(
   color  new_color      // New color
   )
```

Parâmetros

new_color

[in] Nova cor de segundo plano (fundo).

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Angle

Um método "stub".

```
bool  Angle(
   double  angle      // any value
   )
```

Parâmetros

angle

[in] Qualquer valor do tipo double.

Valor de retorno

Sempre falso.

© 2000-2024, MetaQuotes Ltd.

BorderType

Obtém o tipo de borda.

```
int  BorderType() const
```

Valor de retorno

Tipo de borda do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

BorderType

Define o tipo de borda.

```
bool  BorderType(
   int  type      // Border type
   )
```

Parâmetros

type

[in] Novo tipo de borda.

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo objeto gráfico.

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo objeto (OBJ_RECTANGLE_LABEL para CChartObjectRectangleLabel).

© 2000-2024, MetaQuotes Ltd.

Gráfico personalizado

Esta secção apresenta ferramentas para trabalhar com gráficos personalizados.

Seu uso simplifica muito a construção de gráficos personalizados, desenhos e visualização de dados.

Classes - desenvolvidas separadamente - para criar objetos gráficos e primitivas para a desenho de vários tipos de diagramas e curvas. Implementada uma variedade de opções de exibição de objetos, isto é: alteração de estilo e cor de linha, sombreamento, trabalho com séries de dados no gráfico, etc.

Classe	Descrição
CCanvas	Classe para facilitar a criação de desenhos personalizados
CChartCanvas	Classe base para implementar classes projetadas a fim de desenhar gráficos e seus elementos
CHistogramChart	Classe para construir histogramas
CLineChart	Classe para desenhar curvas
CPieChart	Classe para construir gráficos de pizza

© 2000-2024, MetaQuotes Ltd.

CCanvas

A classe CCanvas é uma classe para facilitar a criação de gráficos personalizados.

Descrição

A classe CCanvas fornece a criação de um recurso gráfico (com ou sim âncora ao objeto no gráfico) e desenho de primitivas gráficas.

Declaração

```
class CCanvas
```

Cabeçalho

```
#include <Canvas\Canvas.mqh>
```

Hierarquia de herança

CCanvas

Descendentes diretos

CChartCanvas, CFlameCanvas

Métodos de classe por grupos

Geração	
Attach	Vincula o objeto OBJ_BITMAP_LABEL à instância da classe CCanvas
Create	Cria um recurso gráfico sem vinculação ao objeto no gráfico
CreateBitmap	Cria um recurso gráfico que está vinculado ao objeto de gráfico
CreateBitmapLabel	Cria um recurso gráfico que está vinculado ao objeto de gráfico
Destroy	Destrói o recurso gráfico
Propriedades	
ChartObjectName	Recebe o nome do objeto vinculado ao gráfico
ResourceName	Recebe o nome do recurso gráfico
Width	Recebe a largura do recurso gráfico
Height	Recebe a altura do recurso gráfico
LineStyleSet	Define o estilo de linha
Atualização do objeto na tela	
Update	Exibe as alterações na tela
Resize	Redimensiona o recurso gráfico

© 2000-2024, MetaQuotes Ltd.

Geração	
Borracha/preenchimento de cor	
Erase	Limpa ou preenche usando a cor especificada
Acesso a dados	
PixelGet	Obtém a cor do ponto com as coordenadas especificadas
PixelSet	Define a cor do ponto com as coordenadas especificadas
Desenho de primitivas	
LineVertical	Desenha uma linha vertical
LineHorizontal	Desenhar uma linha horizontal
Line	Desenha uma linha de forma livre
Polyline	Desenha uma linha poligonal
Polygon	Desenha um polígono
Rectangle	Desenha um retângulo
Circle	Desenha um círculo
Triangle	Desenha um triângulo
Ellipse	Desenha uma elipse
Arc	Desenha o arco da elipse
Pie	Desenha o sector da elipse
Desenho de primitivas preenchidas	
FillRectangle	Desenha um retângulo preenchido
FillCircle	Desenha um círculo preenchido
FillTriangle	Desenha um triângulo preenchido
FillPolygon	Desenha um polígono preenchido
FillEllipse	Desenha uma elipse preenchida
Fill	Preenche uma área
Desenho de primitivas usando suavização	
PixelSetAA	Desenha um ponto
LineAA	Desenha uma linha
PolylineAA	Desenha uma linha poligonal

© 2000-2024, MetaQuotes Ltd.

Geração	
PolygonAA	Desenha um polígono
TriangleAA	Desenha um triângulo
CircleAA	Desenha um círculo
EllipseAA	Desenha uma elipse
LineWu	Desenha uma linha
PolylineWu	Desenha uma linha poligonal
PolygonWu	Desenha um polígono
TriangleWu	Desenha um triângulo
CircleWu	Desenha um círculo
EllipseWu	Desenha uma elipse
LineThick	Desenha um segmento de uma linha livre com uma espessura predeterminada usando o algoritmo de suavização
LineThickVertical	Desenha um segmento vertical de uma linha livre com uma espessura predeterminada usando o algoritmo de suavização
LineThickHorizontal	Desenha um segmento horizontal de uma linha livre com uma espessura predeterminada usando o algoritmo de suavização
PolygonSmooth	Desenha um polígono com uma espessura predeterminada usando dois algoritmos de suavização
PolygonThick	Desenha um polígono com uma espessura predeterminada usando o algoritmo de suavização
PolylineSmooth	Desenha uma linha poligonal com uma espessura predeterminada usando dos algoritmos de suavização
PolylineThick	Desenha uma linha poligonal com uma espessura predeterminada usando o algoritmo de suavização
Texto	
FontSet	Define o parâmetro de fonte
FontNameSet	Define o nome da fonte
FontSizeSet	Define o tamanho da fonte
FontFlagsSet	Define os sinalizadores da fonte
FontAngleSet	Define o ângulo de inclinação da fonte
FontGet	Obtém os parâmetros de fonte
FontNameGet	Obtém o nome da fonte
FontSizeGet	Obtém o tamanho da fonte

© 2000-2024, MetaQuotes Ltd.

Geração	
FontFlagsGet	Obtém os sinalizadores de fonte
FontAngleGet	Obtém o ângulo de inclinação da fonte
TextOut	Exibe o texto
TextWidth	Obtém a largura do texto
TextHeight	Obtém a altura do texto
TextSize	Obtém o tamanho do texto
Trocar transparência	
TransparentLevelSet	Define o nível de transparência
Entrada/saída	
LoadFromFile	Lê a imagem a partir de um arquivo BMP

© 2000-2024, MetaQuotes Ltd.

Attach

Obtém - a partir do objeto OBJ_BITMAP_LABEL - o recurso gráfico e vincula a ele uma instância da classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identificador de gráf
   const string       objname,                           // nome do objeto
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de processament
   )
```

Cria um recurso gráfico para o objeto OBJ_BITMAP_LABEL e vincula a ele uma instância da classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identificador de gráf
   const string       objname,                           // nome do objeto
   const int          width,                             // largura da imagem em p
   const int          height,                            // altura da imagem em p
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de processament
   )
```

Parâmetros

chart_id

[out] Identificador de gráfico.

objname

[in] Título (nome) do objeto gráfico.

width

[in] Largura de imagem no recurso.

height

[in] Altura de imagem no recurso.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de processamento do canal-alfa. Por padrão, o canal-alfa é ignorado.

Valor de retorno

true se for bem-sucedido, false, se não puder vincular o objeto.

© 2000-2024, MetaQuotes Ltd.

Arc

Desenha o arco de uma elipse inserida no retângulo com cantos (x1,y1) e (x2,y2). Os limites do arco são cortados por linhas do centro da elipse que se estendem para dois pontos com as coordenadas (x3,y3) e (x4,Y4).

```
void  Arc(
   int         x1,      // coordenada X do canto superior esquerdo do retângulo
   int         y1,      // coordenada Y do canto superior esquerdo do retângulo
   int         x2,      // coordenada X do canto inferior direito do retângulo
   int         y2,      // coordenada Y do canto inferior direito do retângulo
   int         x3,      // coordenada X do primeiro ponto para encontrar os limites d
   int         y3,      // coordenada Y do primeiro ponto para encontrar os limites d
   int         x4,      // coordenada X do segundo ponto para encontrar os limites do
   int         y4,      // coordenada Y do segundo ponto para encontrar os limites do
   const uint  clr      // cor
   );
```

Parâmetros

x1

[in] Coordenada X do canto superior esquerdo que define um retângulo.

y1

[in] Coordenada Y do canto superior esquerdo que define um retângulo.

x2

[in] Coordenada X do canto inferior direito que define um retângulo.

y2

[in] Coordenada Y do canto inferior direito que define um retângulo.

x3

[in] Coordenada X do primeiro ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

y3

[in] Coordenada Y do primeiro ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

x4

[in] Coordenada X do segundo ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

y4

[in] Coordenada Y do segundo ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

clr

[in] Cor no formato ARGB. Para modificar a cor para o formato ARGB, use a função ColorToARGB().

© 2000-2024, MetaQuotes Ltd.

Desenha o arco de uma elipse centrado em (x,y), inscrita num retângulo com raios rx e ry. Os limites do arco são cortados por linhas do centro da elipse determinadas pelos ângulos fi3 e fi4.

```
void  Arc(
   int         x,       // coordenada X do centro da elipse
   int         y,       // coordenada Y do centro da elipse
   int         rx,      // raio da elipse na coordenada X
   int         ry,      // raio da elipse na coordenada Y
   int         fi3,     // ângulo - da linha que parte do centro da elipse - que defin
   int         fi4,     // ângulo - da linha que parte do centro da elipse - que defin
   const uint  clr      // cor
   );
```

Desenha o arco de uma elipse centrado em (x,y), inscrito num retângulo com raios rx e ry, e retorna as coordenadas dos limites do arco. Os limites do arco são cortados por linhas do centro da elipse determinadas pelos ângulos fi3 e fi4.

```
void  Arc(
   int         x,       // coordenada X do centro da elipse
   int         y,       // coordenada Y do centro da elipse
   int         rx,      // raio da elipse na coordenada X
   int         ry,      // raio da elipse na coordenada Y
   int         fi3,     // ângulo - da linha que parte do centro da elipse - que defin
   int         fi4,     // ângulo - da linha que parte do centro da elipse - que defin
   int&        x3,      // coordenada X do primeiro limite do arco
   int&        y3,      // coordenada Y do primeiro limite do arco
   int&        x4,      // coordenada X do segundo limite do arco
   int&        y4,      // coordenada Y do segundo limite do arco
   const uint  clr      // cor
   );
```

Parâmetros

x

[in] Coordenada X do centro da elipse.

y

[in] Coordenada Y do centro da elipse.

rx

[in] Raio da elipse na coordenada X, em pixels.

ry

[in] Raio da elipse na coordenada Y, em pixels.

fi3

[in] Ângulo em radianos que define o primeiro limite do arco

fi4

© 2000-2024, MetaQuotes Ltd.

[in] Ângulo em radianos que define o segundo limite do arco

x3

[out] Variável para obter a coordenada X do primeiro limite do arco.

y3

[out] Variável para obter a coordenada Y do segundo limite do arco.

x4

[out] Variável para obter a coordenada X do segundo limite do arco.

y4

[out] Variável para obter a coordenada Y do segundo limite do arco.

clr

[in] Cor no formato ARGB. Para modificar a cor para o formato ARGB, use a função ColorToARGB().

Exemplos de chamadas de métodos de classe:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- create canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- clear canvas
   canvas.Erase(clrWhite);
//--- draw rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- draw first arc
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- draw second arc
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- print coordinates of arc
   PrintFormat("First point of arc at (%G,%G), second point of arc at (%G,%G)",x1,y1,x
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- show updated canvas
   canvas.Update();
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Pie

Desenha o sector de uma elipse inserida no retângulo com cantos em (x1,y1) e (x2,y2). Os limites do sector são cortados por linhas do centro da elipse que se estendem para dois pontos com as coordenadas (x3,y3) e (x4,Y4).

```
void  Pie(
   int         x1,      // coordenada X do canto superior esquerdo do retângulo
   int         y1,      // coordenada Y do canto superior esquerdo do retângulo
   int         x2,      // coordenada X do canto inferior direito do retângulo
   int         y2,      // coordenada Y do canto inferior direito do retângulo
   int         x3,      // coordenada X do primeiro ponto para encontrar os limites d
   int         y3,      // coordenada Y do primeiro ponto para encontrar os limites d
   int         x4,      // coordenada X do segundo ponto para encontrar os limites do
   int         y4,      // coordenada Y do segundo ponto para encontrar os limites do
   const uint  clr,     // cor de linha
   const uint  fill_clr // cor de preenchimento
   );
```

Parâmetros

x1

[in] Coordenada X do canto superior esquerdo que define um retângulo.

y1

[in] Coordenada Y do canto superior esquerdo que define um retângulo.

x2

[in] Coordenada X do canto inferior direito que define um retângulo.

y2

[in] Coordenada Y do canto inferior direito que define um retângulo.

x3

[in] Coordenada X do primeiro ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

y3

[in] Coordenada Y do primeiro ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

x4

[in] Coordenada X do segundo ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

y4

[in] Coordenada Y do segundo ponto até ao qual é traçada a linha do centro do retângulo para encontrar os limites do arco.

clr

[in] Cor de borda do sector em formato ARGB.

fill_clr

© 2000-2024, MetaQuotes Ltd.

[in] Cor de preenchimento do sector em formato ARGB. Para modificar a cor para o formato ARGB, use a função ColorToARGB().

Desenha o sector preenchido de uma elipse centrado em (x,y), inscrita num retângulo com raios rx e ry. Os limites do sector são cortados por linhas do centro da elipse determinadas pelos ângulos fi3 e fi4.

```
void  Pie(
   int         x,       // coordenada X do centro da elipse
   int         y,       // coordenada Y do centro da elipse
   int         rx,      // raio da elipse na coordenada X
   int         ry,      // raio da elipse na coordenada Y
   int         fi3,     // ângulo - da linha que parte do centro da elipse - que defin
   int         fi4,     // ângulo - da linha que parte do centro da elipse - que defin
   const uint  clr,     // cor de linha
   const uint  fill_clr // cor de preenchimento
   );
```

Desenha o sector preenchido de uma elipse centrado em (x,y), inscrita num retângulo com raios rx e ry, e retorna as coordenadas dos limites do arco. Os limites do sector são cortados por linhas do centro da elipse determinadas pelos ângulos fi3 e fi4.

```
void  Pie(
   int         x,       // coordenada X do centro da elipse
   int         y,       // coordenada Y do centro da elipse
   int         rx,      // raio da elipse na coordenada X
   int         ry,      // raio da elipse na coordenada Y
   int         fi3,     // ângulo - da linha que parte do centro da elipse - que defin
   int         fi4,     // ângulo - da linha que parte do centro da elipse - que defin
   int&        x3,      // coordenada X do primeiro limite do arco
   int&        y3,      // coordenada Y do primeiro limite do arco
   int&        x4,      // coordenada X do segundo limite do arco
   int&        y4,      // coordenada Y do segundo limite do arco
   const uint  clr,     // cor de linha
   const uint  fill_clr // cor de preenchimento
   );
```

Parâmetros

x

[in] Coordenada X do centro da elipse.

y

[in] Coordenada Y do centro da elipse.

rx

[in] Raio da elipse na coordenada X, em pixels.

ry

© 2000-2024, MetaQuotes Ltd.

[in] Raio da elipse na coordenada X, em pixels.

fi3

[in] Ângulo em radianos que define o primeiro limite do arco

fi4

[in] Ângulo em radianos que define o segundo limite do arco

x3

[out] Variável para obter a coordenada X do primeiro limite do arco.

y3

[out] Variável para obter a coordenada Y do segundo limite do arco.

x4

[out] Variável para obter a coordenada X do segundo limite do arco.

y4

[out] Variável para obter a coordenada Y do segundo limite do arco.

clr

[in] Cor de borda do sector em formato ARGB.

fill_clr

[in] Cor de preenchimento do sector em formato ARGB. Para modificar a cor para o formato ARGB, use a função ColorToARGB().

Exemplos de chamadas de métodos de classe:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- create canvas
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Error creating canvas: ",GetLastError());
     }
//--- clear canvas
   canvas.Erase(clrWhite);
//--- draw rectangle
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- draw first pie
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- draw second pie
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Colo
//--- show updated canvas
```

© 2000-2024, MetaQuotes Ltd.

```
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2024, MetaQuotes Ltd.

FillPolygon

Desenha um polígono preenchido.

```
void  FillPolygon(
   int&          x,       // matriz com coordenadas X dos pontos do polígono
   int&          y,       // matriz com coordenadas Y dos pontos do polígono
   const uint   clr       // cor
   );
```

Parâmetros

x

[in] Matriz que contém as coordenadas X dos pontos do polígono.

y

[in] Matriz que contém as coordenadas Y dos pontos do polígono.

clr

[in] Cor no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

FillEllipse

Desenha uma elipse preenchida inscrita num retângulo com as coordenadas especificadas.

```
void  FillPolygon(
   int         x1,      // coordenada X do canto superior esquerdo do retângulo
   int         y1,      // coordenada Y do canto superior esquerdo do retângulo
   int         x2,      // coordenada X do canto inferior direito do retângulo
   int         y2,      // coordenada Y do canto inferior direito do retângulo
   const uint  clr       // cor da elipse
   );
```

Parâmetros

x1

[in] Coordenada X do canto superior esquerdo que define um retângulo.

y1

[in] Coordenada Y do canto superior esquerdo que define um retângulo.

x2

[in] Coordenada X do canto inferior direito que define um retângulo.

y2

[in] Coordenada Y do canto inferior direito que define um retângulo.

clr

[in] Cor no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

GetDefaultColor

Retorna uma cor pré-definida de acordo com seu índice.

```
static uint  GetDefaultColor(
   const uint  i      // índice
   );
```

Parâmetros

i

[in] Índice para obter a cor.

Valor de retorno

Cor.

© 2000-2024, MetaQuotes Ltd.

ChartObjectName

Recebe o nome de um objeto gráfico vinculado.

```
string  ChartObjectName();
```

Valor de retorno

O nome de um objeto gráfico vinculado

© 2000-2024, MetaQuotes Ltd.

Circle

Desenha um círculo

```
void  Circle(
   int         x,        // X coordinate
   int         y,        // Y coordinate
   int         r,        // radius
   const uint  clr       // color
   );
```

Parâmetros

x

[in] coordenada X do centro do círculo.

y

[in] coordenada y do centro do círculo.

r

[in] Raio do círculo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

CircleAA

Desenha um círculo usando um algoritmo de suavização.

```
void  CircleAA(
   const int     x,        // X coordinate
   const int     y,        // Y coordinate
   const double  r,        // radius
   const uint    clr       // color
   );
```

Parâmetros

x

[in] coordenada X do centro do círculo.

y

[in] coordenada y do centro do círculo.

r

[in] Raio do círculo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

CircleWu

Desenha um círculo usando um algoritmo de suavização Wu.

```
void  CircleWu(
   const int     x,        // X coordinate
   const int     y,        // Y coordinate
   const double  r,        // radius
   const uint    clr       // color
   );
```

Parâmetros

x

[in] coordenada X do centro do círculo.

y

[in] coordenada y do centro do círculo.

r

[in] Raio do círculo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

Create

Cria um recurso gráfico sem ligar a um objeto gráfico.

```
virtual bool  Create(
   const string       name,                                  // name
   const int          width,                                 // width
   const int          height,                                // height
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Parâmetros

name

[in] Base para o nome do recurso gráfico. Um nome de recurso é gerado durante a criação, adicionando uma seqüência pseudo aleatória.

width

[in] Largura (tamanho ao longo do eixo X) em pixels.

height

[in] Altura (tamanho ao longo do eixo Y) em pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de processamento de cores. Veja a descrição da função ResourceCreate() para saber mais sobre os métodos de processamento de cores.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

CreateBitmap

Cria um recurso gráfico vinculado a um objeto gráfico.

1. Cria um recurso gráfico na janela principal do gráfico corrente.

```
bool  CreateBitmap(
   const string       name,                                  // name
   const datetime     time,                                  // time
   const double       price,                                 // price
   const int          width,                                 // width
   const int          height,                                // height
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

2. Cria um recurso gráfico, usando um ID de gráfico e um número da sub-janela.

```
bool  CreateBitmap(
   const long         chart_id,                              // chart ID
   const int          subwin,                                // subwindow number
   const string       name,                                  // name
   const datetime     time,                                  // time
   const double       price,                                 // price
   const int          width,                                 // width
   const int          height,                                // height
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Parâmetros

chart_id

[in] ID Chart para criar um objeto.

subwin

[in] Número do Gráfico na sub-janela para criar um objeto.

name

[in] Nome do objeto gráfico e uma base para um nome de recurso gráfico.

time

[in] Ancoragem de objeto gráfico com coordenada de tempo.

price

[in] Ancoragem de objeto gráfico com coordenada de preço.

width

[in] Recurso gráfico da largura (tamanho ao longo do eixo X) em pixels.

height

[in] Recurso gráfico da largura (tamanho ao longo do eixo Y) em pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2024, MetaQuotes Ltd.

[in] Método de processamento de cores. Veja a descrição da função ResourceCreate() para saber mais sobre os métodos de processamento de cores.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

Observação

Se a primeira versão função for usada, o objeto é criado na janela principal do gráfico corrente.

Tamanho do objeto coincide com o tamanho de um recurso gráfico.

© 2000-2024, MetaQuotes Ltd.

CreateBitmapLabel

Cria um recurso gráfico vinculado a um objeto gráfico.

1. Cria um recurso gráfico na janela principal do gráfico corrente.

```
bool  CreateBitmapLabel(
   const string       name,                                  // name
   const int          x,                                     // X coordinate
   const int          y,                                     // Y coordinate
   const int          width,                                 // width
   const int          height,                                // height
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

2. Cria um recurso gráfico, usando um ID de gráfico e um número da sub-janela.

```
bool  CreateBitmapLabel(
   const long         chart_id,                              // chart ID
   const int          subwin,                                // subwindow number
   const string       name,                                  // name
   const int          x,                                     // X coordinate
   const int          y,                                     // Y coordinate
   const int          width,                                 // width
   const int          height,                                // height
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // format
   );
```

Parâmetros

chart_id

[in] ID Chart para criar um objeto.

subwin

[in] Número do Gráfico na sub-janela para criar um objeto.

name

[in] Nome do objeto gráfico e uma base para um nome de recurso gráfico.

x

[in] Chart object anchor point X coordinate.

y

[in] Chart object anchor point Y coordinate.

width

[in] Recurso gráfico da largura (tamanho ao longo do eixo X) em pixels.

height

[in] Recurso gráfico da largura (tamanho ao longo do eixo Y) em pixels.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2024, MetaQuotes Ltd.

[in] Método de processamento de cores. Veja a descrição da função ResourceCreate() para saber mais sobre os métodos de processamento de cores.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

Observação

Se a primeira versão função for usada, o objeto é criado na janela principal do gráfico corrente.

Tamanho do objeto coincide com o tamanho de um recurso gráfico.

© 2000-2024, MetaQuotes Ltd.

Destroy

Finalidade de destruir o recurso gráfico.

```
void  Destroy();
```

Observação

Se um recurso gráfica é vinculado a um objeto gráfico, o último é excluído.

© 2000-2024, MetaQuotes Ltd.

Ellipse

Desenha uma elipse usando dois pontos.

```
void  Ellipse(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto para formar uma elipse.

y1

[in] Coordenada Y do primeiro ponto para formar uma elipse.

x2

[in] Coordenada X do segundo ponto para formar uma elipse.

y2

[in] Coordenada Y do segundo ponto para formar uma elipse.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

EllipseAA

Desenha uma elipse usando dois pontos e um algoritmo de suavização.

```
void  EllipseAA(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto para formar uma elipse.

y1

[in] Coordenada Y do primeiro ponto para formar uma elipse.

x2

[in] Coordenada X do segundo ponto para formar uma elipse.

y2

[in] Coordenada Y do segundo ponto para formar uma elipse.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

EllipseWu

Desenha uma elipse usando dois pontos e um algoritmo de suavização Wu.

```
void  EllipseWu(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto para formar uma elipse.

y1

[in] Coordenada Y do primeiro ponto para formar uma elipse.

x2

[in] Coordenada X do segundo ponto para formar uma elipse.

y2

[in] Coordenada Y do segundo ponto para formar uma elipse.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

Erase

Finalidade de apagar ou preencher com as cores determinadas.

```
void  Erase(
   const uint  clr=0      // color
   );
```

Parâmetros

clr=0

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

Fill

Preenche uma área.

```
void  Fill(
   int          x,        // X coordinate
   int          y,        // Y coordinate
   const uint   clr       // color
   );
```

Parâmetros

x

[in] coordenada X de preenchimento do ponto de partida.

y

[in] coordenada Y de preenchimento do ponto de partida.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

FillCircle

Desenha um círculo preenchido.

```
void  FillCircle(
   int         x,        // X coordinate
   int         y,        // Y coordinate
   int         r,        // radius
   const uint  clr       // color
   );
```

Parâmetros

x

[in] Coordenada X para preenchimento do círculo.

y

[in] coordenada Y para preenchimento do círculo.

r

[in] Raio do círculo preenchido.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

FillRectangle

Desenha um retângulo preenchido.

```
void  FillRectangle(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto para formar um retângulo.

y1

[in] Coordenada Y do primeiro ponto para formar um retângulo.

x2

[in] Coordenada X do segundo ponto para formar um retângulo.

y2

[in] Coordenada Y do segundo ponto para formar um retângulo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

FillTriangle

Desenha um triângulo preenchido.

```
void  FillTriangle(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   int         x3,      // X coordinate
   int         y3,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro canto do triângulo.

y1

[in] Coordenada Y do primeiro canto do triângulo.

x2

[in] Coordenada X do segundo canto do triângulo.

y2

[in] Coordenada Y do segundo canto do triângulo.

x3

[in] Coordenada X do segundo canto do triângulo.

y3

[in] Coordenada Y do terceiro canto do triângulo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

FontAngleGet

Recebe o ângulo inclinado da fonte.

```
uint  FontAngleGet();
```

Valor de retorno

Ângulo de inclinação da fonte

© 2000-2024, MetaQuotes Ltd.

FontAngleSet

Define o ângulo de inclinação da fonte.

```
bool  FontAngleSet(
   uint  angle      // angle
   );
```

Parâmetros

angle

[in] ângulo de inclinação da fonte em décimos de um grau.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

FontFlagsGet

Recebe a fonte das flags.

```
uint  FontFlagsGet();
```

Valor de retorno

font flags

© 2000-2024, MetaQuotes Ltd.

FontFlagsSet

Configura as fontes flags

```
bool  FontFlagsSet(
   uint  flags      // flags
   );
```

Parâmetros

flags

[in] Font creation flags. Veja a descrição da função TextSetFont() para saber mais sobre as flags.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

FontGet

Recebe os parâmetros de fonte atuais.

```
void  FontGet(
   string&  name,       // name
   int&     size,       // size
   uint&    flags,      // flags
   uint&    angle       // slope angle
   );
```

Parâmetros

name

[out] Referência da variável para o retorno do nome da fonte.

size

[out] Referência da variável para o retorno do tamanho da fonte.

flags

[out] Referência da variável para o retorno da flag da fonte.

angle

[out] Referência da variável para retornar um ângulo de inclinação da fonte.

© 2000-2024, MetaQuotes Ltd.

FontNameGet

Recebe o nome da fonte.

```
string  FontNameGet();
```

Valor de retorno

font name

© 2000-2024, MetaQuotes Ltd.

FontNameSet

Configura o nome da fonte.

```
bool  FontNameSet(
   string  name       // nome
   );
```

Parâmetros

name

[in] Font name. Por exemplo, "Arial".

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

FontSet

Define a fonte atual.

```
bool  FontSet(
   const string  name,         // name
   const int     size,         // size
   const uint    flags=0,      // flags
   const uint    angle=0       // angle
   );
```

Parâmetros

name

[in] Font name. Por exemplo, "Arial".

size

[in] Font size. Veja a descrição da função TextSetFont() para aprender mais sobre como configurar o tamanho.

flags=0

[in] Font creation flags. Veja a descrição da função TextSetFont() para saber mais sobre as flags.

angle=0

[in] ângulo de inclinação da fonte em décimos de um grau.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

FontSizeGet

Recebe o tamanho da fonte.

```
int  FontSizeGet();
```

Valor de retorno

font size

© 2000-2024, MetaQuotes Ltd.

FontSizeSet

Configura o tamanho da fonte

```
bool  FontSizeSet(
   int  size      // size
   );
```

Parâmetros

size

[in] Font size. Veja a descrição da função TextSetFont() para aprender mais sobre como configurar o tamanho.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Height

Recebe o comprimento de um recurso gráfico.

```
int  Height();
```

Valor de retorno

comprimento de um recurso gráfico

© 2000-2024, MetaQuotes Ltd.

Line

Desenha um segmento de uma linha à mão livre.

```
void  Line(
   int         x1,       // X coordinate
   int         y1,       // Y coordinate
   int         x2,       // X coordinate
   int         y2,       // Y coordinate
   const uint  clr       // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

LineAA

Desenha um segmento de uma linha à mão livre usando o algoritmo de suavização de borda.

```
void  LineAA(
   const int   x1,                 // X coordinate
   const int   y1,                 // Y coordinate
   const int   x2,                 // X coordinate
   const int   y2,                 // Y coordinate
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

LineWu

Desenha um segmento de uma linha à mão livre usando o algoritmo de suavização Wu.

```
void  LineWu(
   const int   x1,                 // X coordinate
   const int   y1,                 // Y coordinate
   const int   x2,                 // X coordinate
   const int   y2,                 // Y coordinate
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

LineHorizontal

Desenha um segmento de linha horizontal.

```
void  LineHorizontal(
   int         x1,       // X coordinate
   int         x2,       // X coordinate
   int         y,        // Y coordinate
   const uint  clr       // color
   );
```

Parâmetros

x1

[in] coordenada X do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y

[in] Coordenada Y do segmento.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

LineVertical

Desenha um segmento de linha horizontal.

```
void  LineVertical(
   int         x,       // X coordinate
   int         y1,      // Y coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x

[in] coordenada X do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

LineStyleSet

Configura o estilo de linha.

```
void  LineStyleSet(
   const uint  style      // style
   );
```

Parâmetros

style

[in] Line style.

Observação

O parâmetro de entrada pode ser qualquer um dos valores da enumeração ENUM_LINE_STYLE. Além disso, é possível criar um estilo de desenho de linha personalizada.

© 2000-2024, MetaQuotes Ltd.

LineThick

Desenha um segmento de uma linha livre com uma espessura predeterminada usando o algoritmo de suavização.

```
void  LineThick(
   const int      x1,            // coordenada X do primeiro ponto do segmento
   const int      y1,            //coordenada Y do primeiro ponto do segmento
   const int      x2,            // coordenada X do segundo ponto do segmento
   const int      y2,            // coordenada Y do segundo ponto do segmento
   const uint     clr,           // cor
   const int      size,          // espessura da linha
   const uint     style,         // estilo de linha
   ENUM_LINE_END  end_style      // estilo de extremidades de linha
   )
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END

ENUM_LINE_END

Identificador	Descrição
LINE_END_ROUND	As extremidades das linhas são arredondados.
LINE_END_BUTT	As extremidades das linhas são recortadas.

© 2000-2024, MetaQuotes Ltd.

Identificador	Descrição
LINE_END_SQUARE	A linha termina com um retângulo preenchido.

© 2000-2024, MetaQuotes Ltd.

LineThickVertical

Desenha um segmento vertical de uma linha livre com uma espessura predeterminada usando o algoritmo de suavização.

```
void  LineThickVertical(
   const int      x,             // coordenada X do segmento
   const int      y1,            //coordenada Y do primeiro ponto do segmento
   const int      y2,            // coordenada Y do segundo ponto do segmento
   const uint     clr,           // cor
   const int      size,          // espessura da linha
   const uint     style,         // estilo de linha
   ENUM_LINE_END  end_style      // estilo de extremidades de linha
   )
```

Parâmetros

x

[in] Coordenada X do segmento.

y1

[in] Coordenada Y do primeiro ponto do segmento.

y2

[in] Coordenada Y do segundo ponto do segmento.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

LineThickHorizontal

Desenha o segmento horizontal de uma linha livre de espessura predeterminada com suavização

```
void  LineThickHorizontal(
   const int      x1,            // coordenada X do primeiro ponto do segmento
   const int      x2,            // coordenada X do segundo ponto do segmento
   const int      y,             // coordenada Y do segmento
   const uint     clr,           // cor
   const int      size,          // espessura da linha
   const uint     style,         // estilo de linha
   ENUM_LINE_END  end_style      // estilo de extremidades de linha
   )
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto do segmento.

x2

[in] Coordenada X do segundo ponto do segmento.

y

[in] Coordenada Y do segmento.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

LoadFromFile

Lê uma imagem de um arquivo BMP

```
bool  LoadFromFile(
   const string  filename      // file name
   );
```

Parâmetros

filename

[in] Nome do arquivo (incluindo a extensão "BMP").

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

PixelGet

Recebe a cor do ponto com coordenadas específicas.

```
uint  PixelGet(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   );
```

Parâmetros

x

[in] Coordenada do ponto X.

y

[in] Coordenada do ponto Y.

Valor de retorno

Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

PixelSet

Configura a cor do ponto com coordenadas específicas.

```
void  PixelSet(
   const int   x,        // X coordinate
   const int   y,        // Y coordinate
   const uint  clr       // color
   );
```

Parâmetros

x

[in] Coordenada do ponto X.

y

[in] Coordenada do ponto Y.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

PixelSetAA

Desenha um círculo usando o algorítmo de suavização de borda

```
void  PixelSetAA(
   const double  x,        // X coordinate
   const double  y,        // Y coordinate
   const uint    clr       // color
   );
```

Parâmetros

x

[in] Coordenada do ponto X.

y

[in] Coordenada do ponto Y.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

Polygon

Desenha um polígono.

```
void  Polygon(
   int&        x[],     // array of X coordinates
   int&        y[],     // array of Y coordinates
   const uint  clr      // color
   );
```

Parâmetros

x[]

[in] Array de coordenadas X dos pontos de um polígono.

y[]

[in] Array de coordenadas Y dos pontos de um polígono.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

PolygonAA

Desenha um polígno usando o algorítmo de suavização.

```
void  PolygonAA(
   int&        x[],                // array of X coordinates
   int&        y[],                // array of Y coordinates
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x[]

[in] Array de coordenadas X dos pontos de um polígono.

y[]

[in] Array de coordenadas Y dos pontos de um polígono.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

PolygonWu

Desenha um polígno usando o algorítmo de suavização Wu.

```
void  PolygonWu(
   int&        x[],                // array of X coordinates
   int&        y[],                // array of Y coordinates
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x[]

[in] Array de coordenadas X dos pontos de um polígono.

y[]

[in] Array de coordenadas Y dos pontos de um polígono.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

PolygonThick

Desenha um polígono com uma espessura predeterminada usando o algoritmo de suavização.

```
void  PolygonThick(
   const int&      x[],          // matriz de coordenadas X de pontos de polígono
   const int&      y[],          // matriz de coordenadas Y de pontos de polígono
   const uint      clr,          // cor
   const int       size,         // espessura da linha
   const uint      style,        // estilo de linha
   ENUM_LINE_END   end_style     // estilo de extremidades de linha
   )
```

Parâmetros

x[]

[in] Matriz de coordenadas X de pontos de polígono.

y[]

[in] Matriz de coordenadas Y de pontos de polígono.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END.

© 2000-2024, MetaQuotes Ltd.

PolygonSmooth

Desenha um polígono com uma espessura predeterminada usando dois algoritmos de suavização sequencialmente. Em primeiro lugar, com base nas curvas de Bezier, são suavizados segmentos individuais. Em seguida, para melhorar a qualidade da plotagem em relação ao polígono construído a partir destes segmentos, é aplicado o algoritmo bitmap de suavização

```
void  PolygonSmooth(
   int&             x[],                        // matriz de coordenadas X de pontos
   int&             y[],                        // matriz de coordenadas Y de pontos
   const uint       clr,                        // cor
   const int        size,                       // espessura de linha
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // estilo de linha
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // estilo de estremidades de linha
   double           tension=0.5,                // valor do parâmetro de suavização
   double           step=10                     // comprimento das linhas de aproxi
   )
```

Parâmetros

&x[]

[in] Matriz de coordenadas X de pontos de polígono.

&y[]

[in] Matriz de coordenadas Y de pontos de polígono.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style=STYLE_SOLID

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style=LINE_END_ROUND

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END.

tension=0.5

[in] Valor do parâmetro de suavização.

step=10

[in] Comprimento das linhas que se aproximam.

© 2000-2024, MetaQuotes Ltd.

Polyline

Desenha várias linhas.

```
void  Polyline(
   int&        x[],      // array of X coordinates
   int&        y[],      // array of Y coordinates
   const uint  clr       // color
   );
```

Parâmetros

x[]

[in] Array de coordenadas X de várias linhas.

y[]

[in] Array de coordenadas Y de várias linhas.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

PolylineSmooth

Desenha uma linha poligonal com uma espessura predeterminada usando dois algoritmos de suavização sequencialmente. Em primeiro lugar, com base nas curvas de Bezier, são suavizados segmentos individuais da linha. Em seguida, para melhorar a qualidade da plotagem em relação à linha poligonal construída a partir destes segmentos, é aplicado o algoritmo bitmap de suavização

```
void  PolylineSmooth(
   const int&       x[],                        // matriz de coordenadas X de pontos
   const int&       y[],                        // matriz de coordenadas Y de pontos
   const uint       clr,                        // cor
   const int        size,                       // espessura de linha
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // estilo de linha
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // estilo de estremidades de linha
   double           tension=0.5,                // valores de parâmetro de suavizaçã
   double           step=10                     // passo de aproximação
   )
```

Parâmetros

&x[]

[in] Matriz de coordenadas X de pontos de linha poligonal.

&y[]

[in] Matriz de coordenadas Y de pontos de linha poligonal.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style=STYLE_SOLID

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style=LINE_END_ROUND

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END.

tension=0.5

[in] Valor do parâmetro de suavização.

step=10

[in] Passo de aproximação.

© 2000-2024, MetaQuotes Ltd.

PolylineThick

Desenha uma linha poligonal com uma espessura predeterminada usando o algoritmo de suavização.

```
void  PolylineThick(
   const int      &x[],          // matriz de coordenadas X de pontos de linha poligon
   const int      &y[],          // matriz de coordenadas Y de pontos de linha poligon
   const uint     clr,           // cor
   const int      size,          // espessura da linha
   const uint     style,         // estilo de linha
   ENUM_LINE_END  end_style      // estilo de extremidades de linha
   )
```

Parâmetros

&x[]

[in] Matriz de coordenadas X de pontos de linha poligonal.

&y[]

[in] Matriz de coordenadas Y de pontos de linha poligonal.

clr

[in] Cor no formato ARGB.

size

[in] Espessura de linha.

style

[in] Estilo de linha — um dos valores da enumeração ENUM_LINE_STYLE ou um valor personalizado.

end_style

[in] Estilo de extremidades de linha — um dos valores da enumeração ENUM_LINE_END

© 2000-2024, MetaQuotes Ltd.

PolylineWu

Desenha várias linhas usando o algorítmo de suavização Wu.

```
void  PolylineWu(
   int&        x[],                // array of X coordinates
   int&        y[],                // array of Y coordinates
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x[]

[in] Array de coordenadas X de várias linhas.

y[]

[in] Array de coordenadas Y de várias linhas.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

PolylineAA

Desenha várias linhas usando o algorítmo de suavização.

```
void  PolylineAA(
   int&        x[],                // array of X coordinates
   int&        y[],                // array of Y coordinates
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x[]

[in] Array de coordenadas X de várias linhas.

y[]

[in] Array de coordenadas Y de várias linhas.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

Rectangle

Desenha um retângulo usando dois pontos.

```
void  Rectangle(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro ponto para formar um retângulo.

y1

[in] Coordenada Y do primeiro ponto para formar um retângulo.

x2

[in] Coordenada X do segundo ponto para formar um retângulo.

y2

[in] Coordenada Y do segundo ponto para formar um retângulo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

Resize

Redimensiona um recurso gráfico.

```
bool  Resize(
   const int  width,       // width
   const int  height       // height
   );
```

Parâmetros

width

[in] nova largura de um recurso gráfico.

height

[in] novo comprimento de um recurso gráfico.

Valor de retorno

verdade - se bem sucedido, caso contrário - falso.

Observação

Ao redimensionar, a imagem anterior não será salva.

© 2000-2024, MetaQuotes Ltd.

ResourceName

Recebe o nome de um recurso gráfico.

```
string  ResourceName();
```

Valor de retorno

comprimento de um recurso gráfico

© 2000-2024, MetaQuotes Ltd.

TextHeight

Recebe o comprimento do texto.

```
int  TextHeight(
   const string  text      // text
   );
```

Parâmetros

text

[in] Texto para mensurar.

Valor de retorno

comprimento do texto em pixels

Observação

A fonte corrente é utilizada para mensurar o texto.

© 2000-2024, MetaQuotes Ltd.

TextOut

Exibe texto.

```
void  TextOut(
   int         x,              // X coordinate
   int         y,              // Y coordinate
   string      text,           // text
   const uint  clr,            // color
   uint        alignment=0     // alignment
   );
```

Parâmetros

x

[in] coordenada X da âncora do texto.

y

[in] coordenada Y da âncora do texto.

text

[in] Texto a ser exibido.

clr

[in] Cores no formato ARGB.

alignment=0

[in] Método de ancoragem do texto. Veja a descrição da função TextOut() para saber mais sobre métodos de ancoragem.

Observação

A fonte corrente é usada para exibir o texto.

© 2000-2024, MetaQuotes Ltd.

TextSize

Recebe o tamanho do texto.

```
void  TextSize(
   const string  text,        // text
   int&          width,       // width
   int&          height       // height
   );
```

Parâmetros

text

[in] Texto para mensurar.

width

[out] Referência da variável para o retorno da largura da fonte.

height

[out] Referência da variável para o retorno do comprimento do texto.

Observação

A fonte atual é usada para exibir o texto.

© 2000-2024, MetaQuotes Ltd.

TextWidth

Recebe a largura do texto.

```
int  TextWidth(
   const string  text      // text
   );
```

Parâmetros

text

[in] Texto para mensurar.

Valor de retorno

comprimento do texto em pixels

Observação

A fonte atual é usada para exibir o texto.

© 2000-2024, MetaQuotes Ltd.

TransparentLevelSet

Configura o nível de transparência.

```
void  TransparentLevelSet(
   const uchar  value      // value
   );
```

Parâmetros

value

[in] O novo valor do nível de transparência.

Observação

0 significa transparência total, enquanto 255 - para obscuridade total.

A configuração de um nível de transparência afeta tudo o que foi desenhada anteriormente. O nível de transparência especificado não afeta outras construções.

© 2000-2024, MetaQuotes Ltd.

Triângulo

Desenha um triângulo.

```
void  Triangle(
   int         x1,      // X coordinate
   int         y1,      // Y coordinate
   int         x2,      // X coordinate
   int         y2,      // Y coordinate
   int         x3,      // X coordinate
   int         y3,      // Y coordinate
   const uint  clr      // color
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro canto do triângulo.

y1

[in] Coordenada Y do primeiro canto do triângulo.

x2

[in] Coordenada X do segundo canto do triângulo.

y2

[in] Coordenada Y do segundo canto do triângulo.

x3

[in] Coordenada X do segundo canto do triângulo.

y3

[in] Coordenada Y do terceiro canto do triângulo.

clr

[in] Cores no formato ARGB.

© 2000-2024, MetaQuotes Ltd.

TriangleAA

Desenha um triângulo usando o algorítmo de suavização de borda.

```
void  TriangleAA(
   const int   x1,                 // X coordinate
   const int   y1,                 // Y coordinate
   const int   x2,                 // X coordinate
   const int   y2,                 // Y coordinate
   const int   x3,                 // X coordinate
   const int   y3,                 // Y coordinate
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro canto do triângulo.

y1

[in] Coordenada Y do primeiro canto do triângulo.

x2

[in] Coordenada X do segundo canto do triângulo.

y2

[in] Coordenada Y do segundo canto do triângulo.

x3

[in] Coordenada X do segundo canto do triângulo.

y3

[in] Coordenada Y do terceiro canto do triângulo.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

TriangleWu

Desenha um triângulo usando o algorítmo de suavização Wu.

```
void  TriangleWu(
   const int   x1,                 // X coordinate
   const int   y1,                 // Y coordinate
   const int   x2,                 // X coordinate
   const int   y2,                 // Y coordinate
   const int   x3,                 // X coordinate
   const int   y3,                 // Y coordinate
   const uint  clr,                // color
   const uint  style=UINT_MAX      // line style
   );
```

Parâmetros

x1

[in] Coordenada X do primeiro canto do triângulo.

y1

[in] Coordenada Y do primeiro canto do triângulo.

x2

[in] Coordenada X do segundo canto do triângulo.

y2

[in] Coordenada Y do segundo canto do triângulo.

x3

[in] Coordenada X do segundo canto do triângulo.

y3

[in] Coordenada Y do terceiro canto do triângulo.

clr

[in] Cores no formato ARGB.

style=UINT_MAX

[in] O estilo de linha é um dos valores de enumeração ou um valor personalizado ENUM_LINE_STYLE.

© 2000-2024, MetaQuotes Ltd.

Update

Exibe as mudanças na tela.

```
void  Update(
   const bool  redraw=true      // flag
   );
```

Parâmetros

redraw=true

Necessidade do flag de um gráfico redesenhado.

© 2000-2024, MetaQuotes Ltd.

Width

Recebe a largura de um recurso gráfico.

```
int  Width();
```

Valor de retorno

largura do recurso gráfico

© 2000-2024, MetaQuotes Ltd.

CChartCanvas

Classe de base para implementar classes com cuja ajuda ocorre o desenho de gráficos e seus elementos.

Descrição

Esta classe inclui os métodos para lidar com os elementos básicos de todos os gráficos: eixos coordenados e suas marcações, descrições gráficas, grade, fundo, etc. Aqui, são configurados os parâmetros de exibição de elementos: visibilidade/invisibilidade, cor do texto, etc.

Declaração

```
class CChartCanvas : public CCanvas
```

Cabeçalho

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Hierarquia de herança

CCanvas

CChartCanvas

Descendentes diretos

CHistogramChart, CLineChart, CPieChart

Métodos de classe

Método	Ação
ColorBackground	Retorna e define a cor de fundo.
ColorBorder	Retorna e define a cor da borda.
ColorText	Retorna e define a cor do texto.
ColorGrid	Retorna e define a cor da grade.
MaxData	Retorna e define o número máximo de dados permitidos (de séries).
MaxDescrLen	Retorna e define o comprimento máximo dos descritores.
ShowFlags	Retorna e define o sinalizador de visibilidade dos elementos do gráfico.
IsShowLegend	Retorna o sinalizador de visibilidade da legenda do gráfico.
IsShowScaleLeft	Retorna o sinalizador de visibilidade da escala dos valores à esquerda.

© 2000-2024, MetaQuotes Ltd.

Método	Ação
IsShowScaleRight	Retorna o sinalizador de visibilidade da escala dos valores à direita.
IsShowScaleTop	Retorna o sinalizador de visibilidade da escala dos valores acima.
IsShowScaleBottom	Retorna o sinalizador de visibilidade da escala dos valores abaixo.
IsShowGrid	Retorna o sinalizador de visibilidade da grade no gráfico.
IsShowDescriptors	Retorna o sinalizador de visibilidade do descritor no gráfico.
IsShowPercent	Retorna o sinalizador de visibilidade da porcentagem no gráfico.
VScaleMin	Retorna e define o mínimo na escala vertical de valores.
VScaleMax	Retorna e define o máximo na escala vertical de valores.
NumGrid	Retorna e define o número de divisões verticais da escala ao plotar a grade no gráfico.
DataOffset	Retorna e define o valor de deslocamento dos dados.
DataTotal	Retorna o número total de séries de dados no gráfico.
DrawDescriptors	Método virtual para renderizar descritores.
DrawData	Método virtual para desenhar uma série de dados no índice especificado.
Create	Método virtual que cria um recurso gráfico.
AllowedShowFlags	Define o conjunto de sinalizadores de visibilidade permitidos para os elementos no gráfico.
ShowLegend	Define o sinalizador de visibilidade para a legenda.
ShowScaleLeft	Define o sinalizador de visibilidade para a escala esquerda.
ShowScaleRight	Define o sinalizador de visibilidade para a escala direita.

© 2000-2024, MetaQuotes Ltd.

Método	Ação
ShowScaleTop	Define o sinalizador de visibilidade para a escala superior.
ShowScaleBottom	Define o sinalizador de visibilidade para a escala inferior.
ShowGrid	Define o sinalizador de visibilidade para a grade.
ShowDescriptors	Define o sinalizador de visibilidade para os descritores.
ShowValue	Define o sinalizador de visibilidade para os valores.
ShowPercent	Define o sinalizador de visibilidade para as porcentagens.
LegendAlignment	Define o alinhamento do texto para a legenda.
Accumulative	Define os sinalizadores de acumulação de valores para as séries.
VScaleParams	Define os parâmetros para a escala vertical de valores.
DescriptorUpdate	Atualiza o valor do descritor de uma série (numa determinada posição).
ColorUpdate	Atualiza a cor das séries (numa determinada posição).
ValuesCheck	Realiza cálculos internos para plotagem do gráfico.
Redraw	Redesenha o gráfico.
DrawBackground	Desenha o fundo.
DrawLegend	Redesenha a legenda.
DrawLegendVertical	Desenha uma legenda vertical.
DrawLegendHorizontal	Desenha uma legenda horizontal.
CalcScales	Calcula as coordenadas da escala.
DrawScales	Redesenha todas as escalas dos valores.
DrawScaleLeft	Redesenha a escala esquerda dos valores.
DrawScaleRight	Redesenha a escala direita dos valores.
DrawScaleTop	Redesenha a escala superior dos valores.
DrawScaleBottom	Redesenha a escala inferior dos valores.

© 2000-2024, MetaQuotes Ltd.

Método	Ação
DrawGrid	Redesenha a grade.
DrawChart	Redesenha os dados.

Métodos herdados da classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

© 2000-2024, MetaQuotes Ltd.

ColorBackground (método Get)

Retorna a cor de fundo.

```
uint  ColorBackground()
```

Valor de retorno

Cor de fundo.

ColorBackground (método Set)

Define a cor de fundo.

```
void  ColorBackground(
  const uint  value,  // cor de fundo
  )
```

Parâmetros

value

[in] Cor de fundo.

© 2000-2024, MetaQuotes Ltd.

ColorBorder (método Get)

Retorna a cor das bordas.

```
uint  ColorBorder()
```

Valor de retorno

Cor de borda.

ColorBorder (método Set)

Define a cor das bordas.

```
void  ColorBorder(
  const uint  value,  // cor das bordas
  )
```

Parâmetros

value

[in] Cor das bordas.

© 2000-2024, MetaQuotes Ltd.

ColorText (método Get)

Retorna a cor do texto.

```
uint ColorText()
```

Valor de retorno

Cor do texto.

ColorText (método Set)

Define a cor do texto.

```
void ColorText(
  const uint value,  // cor do texto
  )
```

Parâmetros

value

[in] Cor do texto.

© 2000-2024, MetaQuotes Ltd.

ColorGrid (método Get)

Retorna a cor da grade.

```
uint  ColorGrid()
```

Valor de retorno

Cor da grade.

ColorGrid (método Set)

Define a cor da grade.

```
void  ColorGrid(
  const uint  value,  // cor da grade
  )
```

Parâmetros

value

[in] Cor da grade.

© 2000-2024, MetaQuotes Ltd.

MaxData (método Get)

Retorna o número máximo de dados permitidos (de séries).

```
uint  MaxData()
```

Valor de retorno

Número máximo de dados permitidos (de séries).

MaxData (método Set)

Define o número máximo de dados permitidos (de séries).

```
void  MaxData(
  const uint  value,  // número de dados
  )
```

Parâmetros

value

[in] Número máximo de dados (de séries).

© 2000-2024, MetaQuotes Ltd.

MaxDescrLen (método Get)

Retorna o comprimento máximo dos descritores.

```
uint MaxDescrLen()
```

Valor de retorno

Valor do comprimento máximo dos descritores.

MaxDescrLen (método Set)

Define o comprimento máximo dos descritores.

```
void MaxDescrLen(
comprimento máximo
  )
```

Parâmetros

value

[in] Valor do comprimento máximo dos descritores.

© 2000-2024, MetaQuotes Ltd.

ShowFlags (método Get)

Retorna o sinalizador de visibilidade dos elementos do gráfico.

```
bool ShowFlags()
```

Valor de retorno

Valor do sinalizador de visibilidade dos elementos do gráfico.

ShowFlags (método Set)

Define o sinalizador de visibilidade dos elementos do gráfico.

```
void ShowFlags(
  const uint flags,  // sinalizador
  )
```

Parameters

flags

[in] Valor do sinalizador de visibilidade dos elementos do gráfico.

© 2000-2024, MetaQuotes Ltd.

IsShowLegend

Retorna o sinalizador de visibilidade da legenda do gráfico.

```
bool IsShowLegend()
```

Valor de retorno

true, se a legenda é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleLeft

Retorna o sinalizador de visibilidade da escala dos valores à esquerda.

```
bool IsShowScaleLeft()
```

Valor de retorno

true, se a escala de valores é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleRight

Retorna o sinalizador de visibilidade da escala dos valores à direita.

```
bool IsShowScaleRight()
```

Valor de retorno

true, se a escala de valores é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleTop

Retorna o sinalizador de visibilidade da escala dos valores acima.

```
bool IsShowScaleTop()
```

Valor de retorno

true, se a escala de valores é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowScaleBottom

Retorna o sinalizador de visibilidade da escala dos valores abaixo.

```
bool IsShowScaleBottom()
```

Valor de retorno

true, se a escala de valores é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowGrid

Retorna o sinalizador de visibilidade da grade no gráfico.

```
bool  IsShowGrid()
```

Valor de retorno

true, se a grade é visível, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowDescriptors

Retorna o sinalizador de visibilidade do descritor no gráfico.

```
bool IsShowDescriptors()
```

Valor de retorno

true, se os descritores são visíveis, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

IsShowPercent

Retorna o sinalizador de visibilidade da porcentagem no gráfico.

```
bool IsShowPercent()
```

Valor de retorno

true, se as porcentagens são visíveis, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

VScaleMin (método Get)

Retorna o mínimo na escala vertical de valores.

```
double  VScaleMin()
```

Valor de retorno

Valor mínimo na escala vertical.

VScaleMin (método Set)

Define o mínimo na escala vertical de valores.

```
void  VScaleMin(
  const double  value,    // valor na escala vertical
  )
```

Parâmetros

value

[in] Valor mínimo.

© 2000-2024, MetaQuotes Ltd.

VScaleMax

Retorna o máximo na escala vertical de valores.

```
double  VScaleMax()
```

Valor de retorno

Valor máximo na escala vertical.

VScaleMax

Define o máximo na escala vertical de valores.

```
void  VScaleMax(
  const double  value,  //   valor na vertical
  )
```

Parâmetros

value

[in] Valor mínimo.

© 2000-2024, MetaQuotes Ltd.

NumGrid

Retorna o número de divisões verticais da escala ao plotar a grade no gráfico.

```
uint  NumGrid()
```

Valor de retorno

Número de divisões.

NumGrid

Define o número de divisões verticais da escala ao plotar a grade no gráfico.

```
void  NumGrid(
  const uint  value,  // número de divisões
  )
```

Parâmetros

value

[in] Número de divisões.

© 2000-2024, MetaQuotes Ltd.

DataOffset

Retorna o valor de deslocamento dos dados.

```
int  DataOffset()
```

Valor de retorno

Deslocamento de dados.

DataOffset

Define o valor de deslocamento dos dados.

```
void  DataOffset(
  const int  value,  // valor
  )
```

Parâmetros

value

[in] Deslocamento de dados.

© 2000-2024, MetaQuotes Ltd.

DataTotal

Retorna o número total de séries de dados no gráfico.

```
uint  DataTotal()
```

Valor de retorno

Número de ordem de adições.

© 2000-2024, MetaQuotes Ltd.

DrawDescriptors

Método virtual para renderizar descritores.

```
virtual void  DrawDescriptors()
```

© 2000-2024, MetaQuotes Ltd.

DrawData

Método virtual para desenhar uma série de dados no índice especificado.

```
virtual void  DrawData(
   const uint  idx=0,  // índice
   )
```

Parâmetros

idx=0

[in] Índice de série.

© 2000-2024, MetaQuotes Ltd.

Create

Método virtual que cria um recurso gráfico.

```
virtual bool  Create(
  const string       name,       // nome do recurso
  const int          width,      // largura
  const int          height,     // altura
  ENUM_COLOR_FORMAT  clrfmt,     // formato
  )
```

Parâmetros

nome

[in] [in] [in] [in] Base para o nome do recurso gráfico. O nome do recurso é formado ao criar usando a adição de cadeias de caracteres pseudo-aleatórias.

width

[in] Largura (tamanho ao longo do eixo X) em pixels.

height

[in] Altura (tamanho ao longo do eixo Y) em pixels.

clrfmt

[in] Método de processamento de cor. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Valor de retorno

true no caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

AllowedShowFlags

Define o conjunto de sinalizadores de visibilidade permitidos para os elementos no gráfico.

```
void  AllowedShowFlags(
  const uint  flags,  // sinalizadores
  )
```

Parâmetros

flags

[in] Sinalizadores permitidos.

© 2000-2024, MetaQuotes Ltd.

ShowLegend

Define o valor do sinalizador de visibilidade para a legenda (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a legenda torna-se visível.
- false — a legenda torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowScaleLeft

Define o valor do sinalizador de visibilidade para a escala esquerda (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a escala esquerda torna-se visível.
- true — a escala esquerda torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowScaleRight

Define o valor do sinalizador de visibilidade para a escala direita (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a escala direita torna-se visível.
- true — a escala direita torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowScaleTop

Define o valor do sinalizador de visibilidade para a escala superior (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a escala superior torna-se visível.
- true — a escala superior torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowScaleBottom

Define o valor do sinalizador de visibilidade para a escala inferior (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,     // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a escala inferior torna-se visível.
- true — a escala inferior torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowGrid

Define o valor do sinalizador de visibilidade para a grade (FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a grade torna-se visível.
- false — a grade torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowDescriptors

Define o valor do sinalizador de visibilidade para os descritores (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,    // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — o descritor torna-se visível.
- false — o descritor torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowValue

Define o sinalizador de visibilidade para os valores (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — o valor torna-se visível.
- false — o valor torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

ShowPercent

Define o valor do sinalizador de visibilidade para as porcentagens (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // valor do sinalizador
  )
```

Parâmetros

flag

[in] Valor do sinalizador:

- true — a porcentagem torna-se visível.
- false — a porcentagem torna-se invisível.

© 2000-2024, MetaQuotes Ltd.

LegendAlignment

Define o alinhamento do texto para a legenda.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // sinalizador
  )
```

Parâmetros

value

[in] Toma um dos valores da enumeração ENUM_ALIGNMENT:

- ALIGNMENT_LEFT — alinhamento ao longo da borda esquerda.
- ALIGNMENT_TOP — alinhamento ao longo da borda superior.
- ALIGNMENT_RIGHT — alinhamento ao longo da borda direita.
- ALIGNMENT_BOTTOM — alinhamento ao longo da borda inferior.

© 2000-2024, MetaQuotes Ltd.

Accumulative

Define os sinalizadores de acumulação de valores para as séries.

```
void  Accumulative(
  const bool  flag=true,  // valor do sinalizador
  )
```

Parâmetros

flag=true

[in] Valor do sinalizador:

- true — o valor atual da série é substituído pela soma de todos os anteriores.
- false — modo padrão de renderização de séries.

© 2000-2024, MetaQuotes Ltd.

VScaleParams

Define os parâmetros para a escala vertical de valores.

```
void  VScaleParams(
  const double  max,    // máximo
  const double  min,    // mínimo
  const uint    grid,   // número de divisões
  )
```

Parâmetros

max

[in] Valor mínimo.

min

[in] Valor máximo.

grid

[in] Número de divisões da escala.

© 2000-2024, MetaQuotes Ltd.

DescriptorUpdate

Atualiza o valor do descritor de uma série (numa determinada posição).

```
bool  DescriptorUpdate(
  const uint    pos,    // índice
  const string  descr,  // valor
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

descr

[in] Valor do descritor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ColorUpdate

Atualiza a cor das séries (numa determinada posição).

```
bool  ColorUpdate(
  const uint  pos,  // índice
  const uint  clr,  // cor
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

clr

[in] Valor de cor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValuesCheck

Método virtual auxiliar que realiza cálculos internos para plotagem do gráfico.

```
virtual void  ValuesCheck()
```

© 2000-2024, MetaQuotes Ltd.

Redraw

Método virtual para redesenhar o gráfico.

```
virtual void  Redraw()
```

© 2000-2024, MetaQuotes Ltd.

DrawBackground

Método virtual para redesenhar o fundo.

```
virtual void  DrawBackground()
```

© 2000-2024, MetaQuotes Ltd.

DrawLegend

Método virtual para redesenhar a legenda.

```
virtual void  DrawLegend()
```

© 2000-2024, MetaQuotes Ltd.

DrawLegendVertical

Desenha uma legenda vertical.

```
int  DrawLegendVertical(
  const int  w,  // largura
  const int  h,  // altura
  )
```

Parâmetros

w

[in] Largura máxima do texto na legenda.

h

[in] Altura máxima do texto na legenda.

Valor de retorno

Largura da legenda em pixels.

© 2000-2024, MetaQuotes Ltd.

DrawLegendHorizontal

Desenha uma legenda horizontal.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Parâmetros

w

[in] Largura máxima do texto na legenda.

h

[in] Altura máxima do texto na legenda.

Valor de retorno

Altura da legenda em pixels.

© 2000-2024, MetaQuotes Ltd.

CalcScales

Método virtual para calcular as coordenadas dos rótulos para a escala de valores.

```
virtual void  CalcScales()
```

© 2000-2024, MetaQuotes Ltd.

DrawScales

Método virtual para redesenhar todas as escalas de valores.

```
virtual void  DrawScales()
```

© 2000-2024, MetaQuotes Ltd.

DrawScaleLeft

Método virtual para redesenhar o a escala esquerda de valores.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // sinalizador
  )
```

Parâmetros

draw

[in] Sinalizador que indica se é necessário redesenhar a escala.

Valor de retorno

Largura da escala de valores.

© 2000-2024, MetaQuotes Ltd.

DrawScaleRight

Método virtual para redesenhar o a escala direita de valores.

```
virtual int  DrawScaleRight(
  const bool  draw,  // sinalizador
  )
```

Parâmetros

draw

[in] Sinalizador que indica se é necessário redesenhar a escala.

Valor de retorno

Largura da escala de valores.

© 2000-2024, MetaQuotes Ltd.

DrawScaleTop

Método virtual para redesenhar o a escala superior de valores.

```
virtual int  DrawScaleTop(
  const bool  draw,  // sinalizador
  )
```

Parâmetros

draw

[in] Sinalizador que indica se é necessário redesenhar a escala.

Valor de retorno

Altura da escala de valores.

© 2000-2024, MetaQuotes Ltd.

DrawScaleBottom

Método virtual para redesenhar o a escala inferior de valores.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // sinalizador
  )
```

Parâmetros

draw

[in] Sinalizador que indica se é necessário redesenhar a escala.

Valor de retorno

Altura da escala de valores.

© 2000-2024, MetaQuotes Ltd.

DrawGrid

Método virtual para redesenhar a grade.

```
virtual void  DrawGrid()
```

© 2000-2024, MetaQuotes Ltd.

DrawChart

Método virtual para redesenhar o gráfico.

```
virtual void  DrawChart()
```

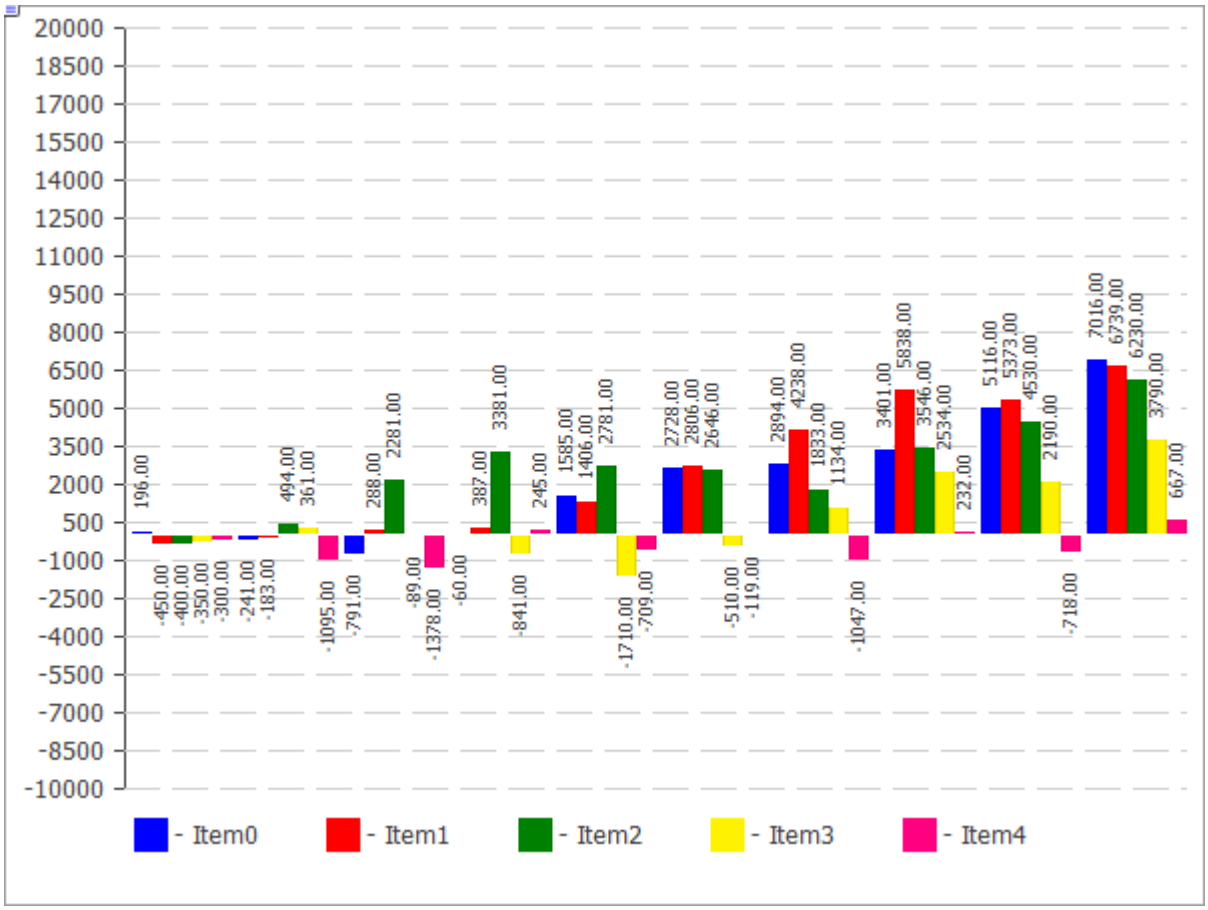
© 2000-2024, MetaQuotes Ltd.

CHistogramChart

Classe para construir histogramas.

Descrição

Nesta classe são implementados todos os métodos de trabalho com construção de histogramas. Com sua ajuda é definida a largura das colunas e configurado o trabalho com grandes quantidades de dados. São incluídos os métodos para trabalhar com preenchimento de gradiente para as colunas dos histogramas, o que permite uma visualização clara dos dados.



O código da imagem acima está abaixo.

Declaração

```
class CHistogramChart : public CChartCanvas
```

Cabeçalho

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Hierarquia de herança

CCanvas
  CChartCanvas
    CHistogramChart

© 2000-2024, MetaQuotes Ltd.

Métodos de classe

Método	Ação
Gradient	Define um sinalizador que indica se as colunas do histograma terão preenchimento de gradiente.
BarGap	Define o valor de recuo a partir da origem das coordenadas do histograma.
BarMinSize	Define a largura mínima das colunas do histograma.
BarBorder	Define um sinalizador que indica se necessário renderizar a borda de cada coluna.
Create	Método virtual que cria um recurso gráfico.
SeriesAdd	Adiciona uma nova série de dados.
SeriesInsert	Insere a série de dados no gráfico.
SeriesUpdate	Atualiza a série de dados no gráfico.
SeriesDelete	Remove a série de dados no gráfico.
ValueUpdate	Atualiza o valor do elemento numa determinada série.
DrawData	Método virtual que desenha um histograma para uma determinada série.
DrawBar	Desenha uma coluna de histograma sob a forma de um retângulo preenchido.
GradientBrush	Cria um pincel para um preenchimento de gradiente.

Métodos herdados da classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos herdados da classe CChartCanvas

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Exemplo

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                    Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                                http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using histogram"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- create chart
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Error creating histogram chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Gradient

Define um sinalizador que indica se as colunas do histograma terão preenchimento de gradiente.

```
void  Gradient(
  const bool  flag=true,  // valor do sinalizador
  )
```

Parâmetros

flag=true

Valor do sinalizador: true, se ativada o preenchimento de gradiente, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

BarGap

Define o valor de recuo a partir da origem das coordenadas do histograma.

```
void  BarGap(
  const uint  value,  // recuo
  )
```

Parâmetros

value

[in] Valor de recudo do histograma.

© 2000-2024, MetaQuotes Ltd.

BarMinSize

Define a largura mínima das colunas do histograma.

```
void  BarMinSize(
  const uint  value,  // largura mínima
  )
```

Parâmetros

value

[in] Largura mínima.

© 2000-2024, MetaQuotes Ltd.

BarBorder

Define um sinalizador que indica se necessário renderizar a borda de cada coluna.

```
void  BarBorder(
  const uint  value,  // sinalizador
  )
```

Parâmetros

value

[in] Valor do sinalizador:

- true — as bordas serão renderizadas
- false — as bordas não serão renderizadas

© 2000-2024, MetaQuotes Ltd.

Create

Método virtual que cria um recurso gráfico.

```
virtual bool  Create(
  const string       name,     // nome
  const int          width,    // largura
  const int          height,   // altura
  ENUM_COLOR_FORMAT  clrfmt,   // formato
  )
```

Parâmetros

nome

[in] [in] [in] [in] Base para o nome do recurso gráfico. O nome do recurso é formado ao criar usando a adição de cadeias de caracteres pseudo-aleatórias.

width

[in] Largura (tamanho ao longo do eixo X) em pixels.

height

[in] Altura (tamanho ao longo do eixo Y) em pixels.

clrfmt

[in] Método de processamento de cor. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Valor de retorno

true no caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesAdd

Adiciona uma nova série de dados.

```
bool  SeriesAdd(
  const double&  value[],  // valores
  const string  descr,     // rótulo
  const uint    clr,       // cor
  )
```

Parâmetros

value[]

[in] Série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesInsert

Insere a série de dados no gráfico.

```
bool  SeriesInsert(
  const uint    pos,        // índice
  const double&  value[],   // valores
  const string  descr,      // rótulo
  const uint    clr,        // cor
  )
```

Parâmetros

pos

[in] Índice para inserção.

value[]

[in] Série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesUpdate

Atualiza a série de dados no gráfico.

```
bool  SeriesUpdate(
  const uint    pos,        // índice
  const double  &value[],   // valores
  const string  descr,      // rótulo
  const uint    clr,        // cor
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

&value[]

[in] Novos valores para a série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesDelete

Remove a série de dados no gráfico.

```
bool  SeriesDelete(
  const uint  pos,  // índice
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Atualiza um determinado valor numa determinada série.

```
bool  ValueUpdate(
  const uint  series,  // índice de série
  const uint  pos,     // índice de elemento
  double      value,   // valor
  )
```

Parâmetros

series

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

pos

[in] Índice de elemento na série.

value

[in] Novo valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

DrawData

Método virtual que desenha um histograma para uma determinada série.

```
virtual void  DrawData(
  const uint  index,  // índice
  )
```

Parâmetros

index

[in] Número de ordem de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

© 2000-2024, MetaQuotes Ltd.

DrawBar

Desenha uma coluna de histograma sob a forma de um retângulo preenchido.

```
void  DrawBar(
  const int   x,    // coordenada x
  const int   y,    // coordenada y
  const int   w,    // largura
  const int   h,    // altura
  const uint  clr,  // cor
  )
```

Parâmetros

x

[in] Coordenada X do ponto esquerdo superior do retângulo.

y

[in] Coordenada Y do ponto esquerdo superior do retângulo.

w

[in] Largura do retângulo.

h

[in] Altura do retângulo.

clr

[in] Cor do retângulo.

© 2000-2024, MetaQuotes Ltd.

GradientBrush

Cria um pincel para um preenchimento de gradiente.

```
void  GradientBrush(
  const int   size,       // tamanho
  const uint  fill_clr,   // cor de preenchimento
  )
```

Parâmetros

size

[in] Espessura de pincel

fill_clr

[in] Cor de preenchimento

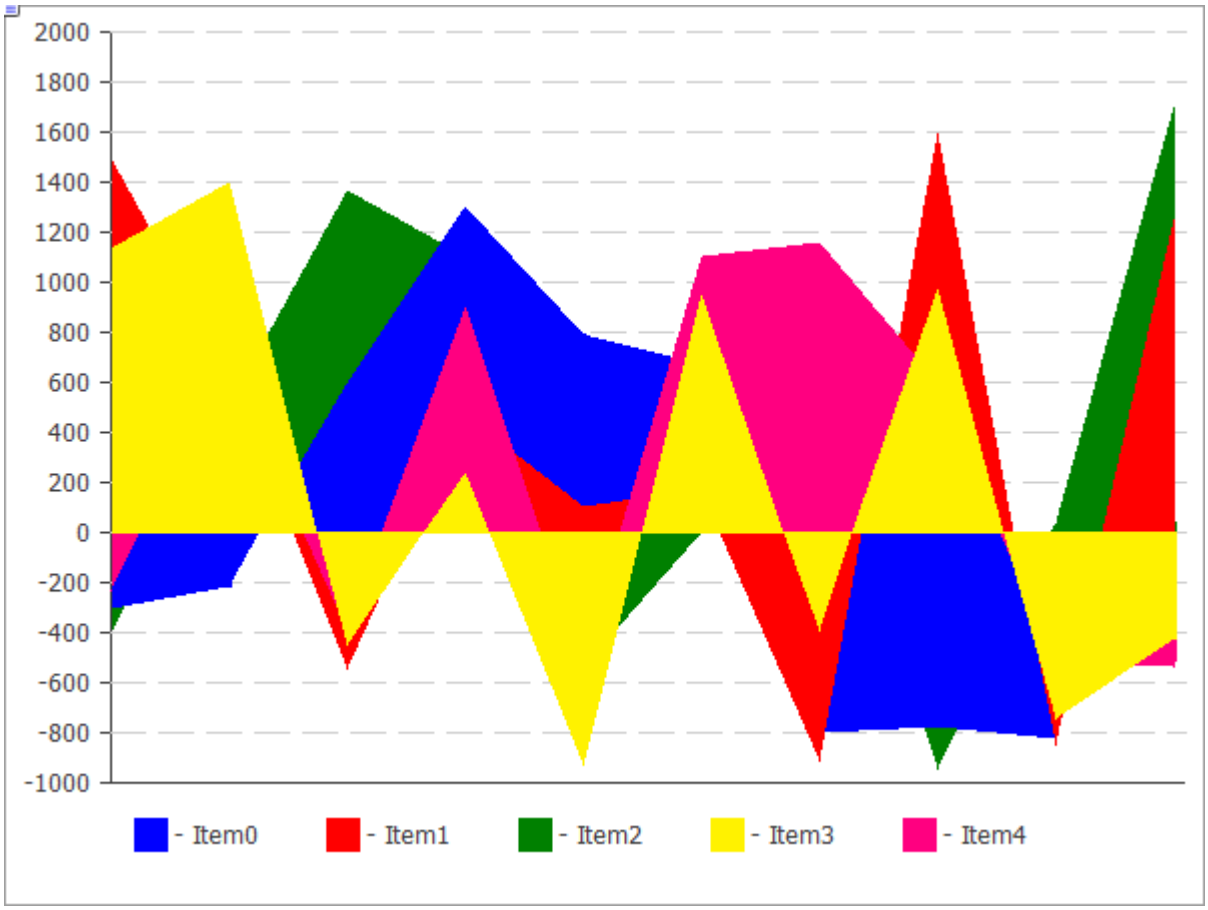
© 2000-2024, MetaQuotes Ltd.

CLineChart

Classe para construir curvas.

Descrição

Os métodos incluídos na classe são projetados para trabalhar com curvas no gráfico. Incluída a possibilidade de preencher uma área delimitada pela curva construída.



O código da imagem acima está abaixo.

Declaração

```
class CLineChart : public CChartCanvas
```

Cabeçalho

```
#include <Canvas\Charts\LineChart.mqh>
```

Hierarquia de herança

CCanvas
  CChartCanvas
    CLineChart

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Method	Action
Filled	Define o sinalizador para preencher uma área sob uma curva definida pela série de dados.
Create	Cria um recurso gráfico.
SeriesAdd	Adiciona uma nova série de dados.
SeriesInsert	Insere a série de dados no gráfico.
SeriesUpdate	Atualiza a série de dados no gráfico.
SeriesDelete	Remove a série de dados no gráfico.
ValueUpdate	Atualiza um determinado valor numa determinada série.
DrawChart	Método virtual que executa a renderização da curva e todos seus elementos.
DrawData	Método virtual que desenha uma curva para uma determinada série.
CalcArea	Calcula a área sob a curva definida pela série de dados.

Métodos herdados da classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos herdados da classe CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Exemplo

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                    Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                                http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using line chart"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- create chart
   CLineChart chart;
//--- create chart
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Error creating line chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- play with values
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- finish
   chart.Destroy();
   return(0);
  }
```

© 2000-2024, MetaQuotes Ltd.

Filled

Define um sinalizador que indica se preciso preencher uma área sob uma curva definida pela série de dados.

```
void  Filled(
  const bool  flag=true,  // sinalizador
  )
```

Parâmetros

flag=true

[in] Valor do sinalizador:

- true — preencher uma área sob a curva
- false — não preencher a área sob a curva

© 2000-2024, MetaQuotes Ltd.

Create

Método virtual que cria um recurso gráfico.

```
virtual bool  Create(
  const string       name,     // nome
  const int          width,    // largura
  const int          height,   // altura
  ENUM_COLOR_FORMAT  clrfmt,   // formato
  )
```

Parâmetros

nome

[in] [in] [in] [in] Base para o nome do recurso gráfico. O nome do recurso é formado ao criar usando a adição de cadeias de caracteres pseudo-aleatórias.

width

[in] Largura (tamanho ao longo do eixo X) em pixels.

height

[in] Altura (tamanho ao longo do eixo Y) em pixels.

clrfmt

[in] Método de processamento de cor. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Valor de retorno

true no caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesAdd

Adiciona uma nova série de dados.

```
bool  SeriesAdd(
  const double&  value[],  // valores
  const string  descr,     // rótulo
  const uint    clr,       // cor
  )
```

Parâmetros

value[]

[in] Série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesInsert

Insere a série de dados no gráfico.

```
bool  SeriesInsert(
  const uint     pos,        // índice
  const double&  value[],    // valores
  const string   descr,      // rótulo
  const uint     clr,        // cor
  )
```

Parâmetros

pos

[in] Índice para inserção.

value[]

[in] Série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesUpdate

Atualiza a série de dados no gráfico.

```
bool  SeriesUpdate(
  const uint     pos,        // índice
  const double&  value[],    // valores
  const string   descr,      // rótulo
  const uint     clr,        // cor
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

value[]

[in] Novos valores para a série de dados.

descr

[in] Rótulo da série.

clr

[in] Cor de exibição da série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesDelete

Remove a série de dados no gráfico.

```
bool  SeriesDelete(
  const uint  pos,  // índice
  )
```

Parâmetros

pos

[in] Número de ordem de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Atualiza um determinado valor numa determinada série.

```
bool  ValueUpdate(
  const uint  series,  // índice de série
  const uint  pos,     // índice de elemento
  double      value,   // valor
  )
```

Parâmetros

series

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

pos

[in] Índice de elemento na série.

value

[in] Novo valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

DrawChart

Método virtual que executa a renderização da curva e todos seus elementos.

```
virtual void  DrawChart()
```

© 2000-2024, MetaQuotes Ltd.

DrawData

Método virtual que desenha uma curva para uma determinada série.

```
virtual void  DrawData(
  const uint  index,  // índice
  )
```

Parameters

index

[in] Número de ordem de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

© 2000-2024, MetaQuotes Ltd.

CalcArea

Calcula a área sob a curva definida pela série de dados.

```
double  CalcArea(
  const uint  index,  // índice
  )
```

Parâmetros

index

[in] Número de ordem de adição de adição de adição de adição de adição de adição - a partir do zero - para o índice de série.

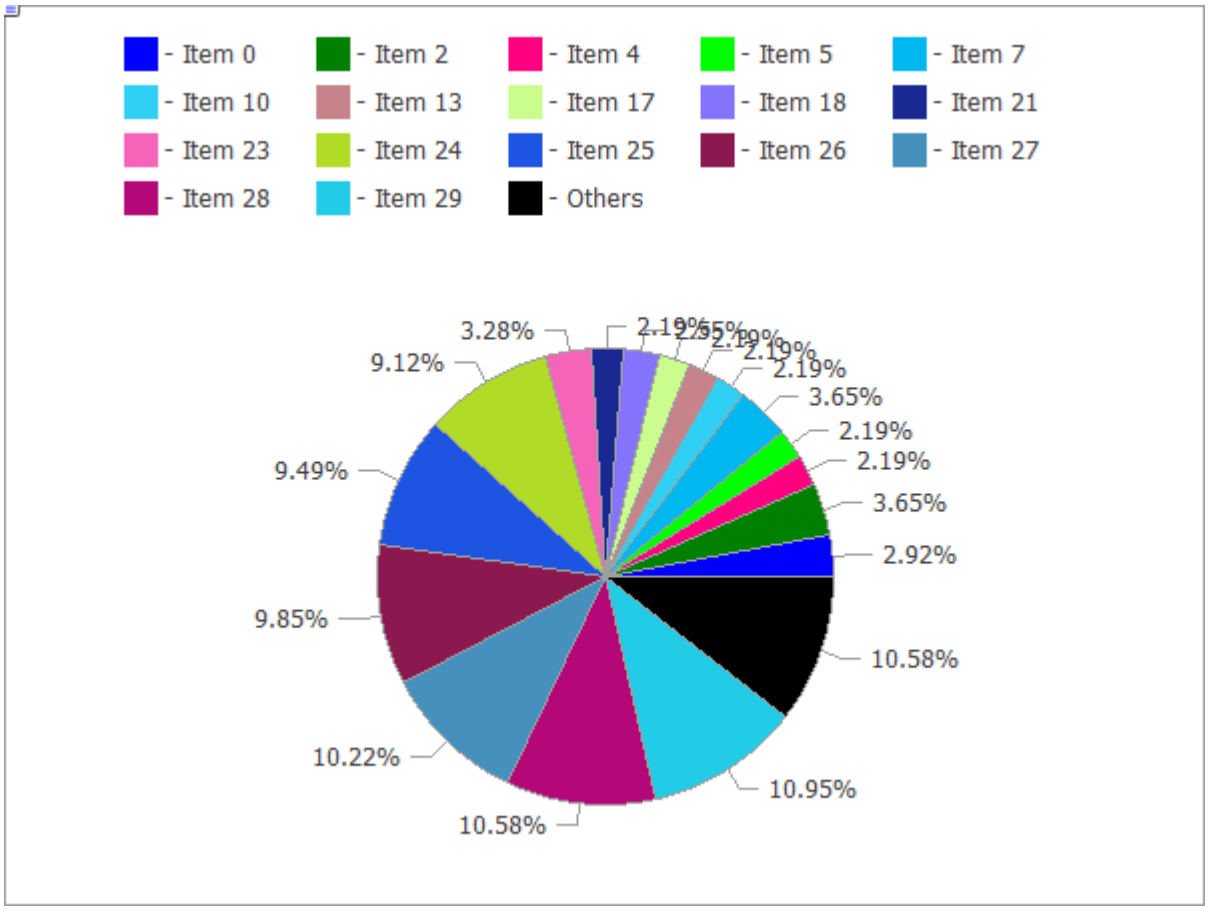
Valor de retorno

Área da forma que é limitada pela curva definida na série de dados.

© 2000-2024, MetaQuotes Ltd.

CPieChart

Classe para construir gráficos de pizza.



O código da imagem acima está abaixo.

Descrição

Os métodos incluídos nesta classe são projetados para trabalhar em tempo integral com o gráfico de pizza, desde a criação do design gráfico até aos textos dos segmentos.

Declaração

```
class CPieChart : public CChartCanvas
```

Cabeçalho

```
#include <Canvas\Charts\PieChart.mqh>
```

Hierarquia de herança

CCanvas
  CChartCanvas
    CPieChart

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Método	Ação
Create	Método virtual que cria um recurso gráfico.
SeriesSet	Define a série de valores que serão apresentados no gráfico.
ValueAdd	Adiciona um novo valor no gráfico (na extremidade).
ValueInsert	Insere um novo valor no gráfico (numa determinada posição).
ValueUpdate	Atualiza o valor no gráfico (numa determinada posição).
ValueDelete	Remove o valor no gráfico (numa determinada posição).
DrawChart	Método virtual que executa a renderização do gráfico e todos seus elementos.
DrawPie	Desenha um segmento - do gráfico - que corresponde a um determinado valor.
LabelMake	Gera um rótulo de segmento com base em seus valores e rótulo original.

Métodos herdados da classe CCanvas

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

Métodos herdados da classe CChartCanvas

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

Exemplo

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
//|                   Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using pie chart"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- check
   if(Width<=0 || Height<=0)
     {
      Print("Too simple.");
      return(-1);
     }
//--- create chart
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
      Print("Error creating pie chart: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- draw
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
//--- disable legend
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
```

```
   Sleep(2000);
//--- disable legend
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- disable percentage
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- disable descriptors
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- enable all
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- or like this
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- play with values
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- play with colors
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- rotate
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- finish
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| Random RGB color                                                 |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Método virtual que cria um recurso gráfico.

```
virtual bool  Create(
  const string       name,     // nome
  const int          width,    // largura
  const int          height,   // altura
  ENUM_COLOR_FORMAT  clrfmt,   // formato
  )
```

Parâmetros

nome

[in] [in] [in] [in] Base para o nome do recurso gráfico. O nome do recurso é formado ao criar usando a adição de cadeias de caracteres pseudo-aleatórias.

width

[in] Largura (tamanho ao longo do eixo X) em pixels.

height

[in] Altura (tamanho ao longo do eixo Y) em pixels.

clrfmt

[in] Método de processamento de cor. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Valor de retorno

true no caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

SeriesSet

Define a série de valores que serão apresentados no gráfico.

```
bool  SeriesSet(
  const double&  value[],  // valores
  const string&  text[],   // rótulos
  const uint&    clr[],    // cores
  )
```

Parâmetros

value[]

[in] Matriz de valores.

text[]

[in] Matriz de rótulos de valores.

clr[]

[in] Matriz de cores de valores.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueAdd

Adiciona um novo valor no gráfico (na extremidade).

```
bool  ValueAdd(
  const double  value,  // valor
  const string  descr,  // rótulo
  const uint    clr,    // cor
  )
```

Parâmetros

value

[in] Valor.

descr

[in] Rótulo de valor.

clr

[in] Cor de valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueInsert

Insere um novo valor no gráfico (numa determinada posição).

```
bool  ValueInsert(
  const uint    pos,    // índice
  const double  value,  // valor
  const string  descr,  // rótulo
  const uint    clr,    // cor
  )
```

Parâmetros

pos

[in] Índice para inserção.

value

[in] Valor.

descr

[in] Rótulo de valor.

clr

[in] Cor de valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueUpdate

Atualiza o valor no gráfico (numa determinada posição).

```
bool  ValueUpdate(
  const uint    pos,     // índice
  const double  value,   // valor
  const string  descr,   // rótulo
  const uint    clr,     // cor
  )
```

Parâmetros

pos

[in] Número de ordem de adição - a partir do zero - para o índice de valor.

value

[in] Valor.

descr

[in] Rótulo de valor.

clr

[in] Cor de valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ValueDelete

Remove o valor no gráfico (numa determinada posição).

```
bool  ValueDelete(
  const uint  pos,  // índice
  )
```

Parâmetros

pos

[in] Número de ordem de adição - a partir do zero - para o índice de valor.

Valor de retorno

true, em caso de sucesso, caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

DrawChart

Método virtual que executa a renderização do gráfico e todos seus elementos.

```
virtual void  DrawChart()
```

© 2000-2024, MetaQuotes Ltd.

DrawPie

Desenha um segmento - do gráfico - que corresponde a um determinado valor.

```
void  DrawPie(
  double      fi3,   // ângulo - do raio que parte do centro do gráfico - que define
  double      fi4,   // ângulo - do raio que parte do centro do gráfico - que define
  int         idx,   // índice
  CPoint&     p[],  //
  const uint  clr,   //
  )
```

Parâmetros

fi3

[in] Ângulo em radianos que define a primeira borda do arco.

fi4

[in] Ângulo em radianos que define a segunda borda do arco.

idx

[in] Índice do valor ao qual corresponde o segmento.

p[]

[in] Matriz de pontos de referência (x, y) para a construção de segmentos.

clr

[in] Cor do segmento.

© 2000-2024, MetaQuotes Ltd.

LabelMake

Gera um rótulo de segmento com base em seus valores e rótulo original.

```
string  LabelMake(
  const string  text,      // rótulo
  const double  value,     // valor
  const bool    to_left,   // sinalizador
  )
```

Parâmetros

text

[in] Rótulo.

value

[in] Valor.

to_left

[in] Define a ordem do rótulo:

- true — rótulo, depois valor.
- false — valor, depois rótulo.

Valor de retorno

Rótulo do segmento.

© 2000-2024, MetaQuotes Ltd.

Gráfico 3D

Esta seção apresenta as classes para a criação de gráficos tridimensionais baseados em funções para trabalhar com o DirectX. Classe base é CCnavas3D e contém métodos de controle de câmera e de iluminação, além disso, fornece um gerenciador de recursos gráficos - texturas, sombreadores, buffers de vértices, índices e parâmetros de sombreadores.

Para começar a trabalhar com a biblioteca, basta ler o artigo Como criar gráfico 3D em DirectX no MetaTrader 5.

Separadamente, são apresentadas classes de objetos básicos de cena, como paralelepípedos, superfícies tridimensionais em dados personalizados ou grades arbitrárias.

© 2000-2024, MetaQuotes Ltd.

CCanvas3D

A classe CCanvas3D é uma classe para criação e visualização simplificadas de objetos tridimensionais em um gráfico.

Descrição

CCanvas3D simplifica bastante a criação e a visualização de grandes quantidades de dados em forma de gráficos 3D animados. A classe contém métodos de controle de câmera e iluminação e também fornece um gerenciador de recursos para a criação de recursos gráficos: texturas, sombreadores, buffers de vértices, índices e parâmetros de sombreador.

Além disso, a biblioteca contém classes de objetos básicos de cena, como paralelepípedos, superfícies tridimensionais em dados personalizados ou grades arbitrárias.

Para que as funções operem, a placa de vídeo deve suportar DX 11 e sombreadores versão 5.0.

Declaração

```
class CCanvas
```

Cabeçalho

```
#include <Canvas\Canvas.mqh>
```

Hierarquia de herança

CCanvas
  CCanvas3D

Métodos de classe por grupos

Criar/excluir	Descrição
Create	Cria um recurso gráfico para renderizar uma cena 3D sem vinculação ao objeto do gráfico.
Attach	Obtém do objeto OBJ_BITMAP_LABEL um recurso gráfico e o vincula a uma instância da classe CCanvas.
ObjectAdd	Adiciona um objeto à cena 3D para posterior renderização.
Destroy	Destrói um recurso gráfico e libera um contexto gráfico 3D.
Trabalho com luz	
AmbientColorSet	Define cor e intensidade de iluminação ambiente não direcional.
AmbientColorGet	Obtém cor e intensidade de iluminação ambiente não direcional.
LightDirectionSet	Especifica a direção da fonte de luz direcional.
LightDirectionGet	Obtém a direção de uma fonte de luz direcional.

© 2000-2024, MetaQuotes Ltd.

Criar/excluir	Descrição
LightColorSet	Define cor e intensidade da fonte de luz direcional.
LightColorGet	Obtém cor e intensidade de uma fonte de luz direcional.
Trabalho com câmera	
ProjectionMatrixSet	Calcula e define a matriz de projeção de uma coordenada 3D em um quadro 2D.
ProjectionMatrixGet	Obtém a matriz de projeção de una cena 3D em um quadro 2D.
ViewMatrixSet	Define a matriz de vista de uma cena 3D.
ViewMatrixGet	Retorna uma matriz de vista de uma cena 3D.
ViewPositionSet	Define a posição do ponto de vista em uma cena 3D.
ViewRotationSet	Define o ponto de enfoque para uma cena 3D.
ViewTargetSet	Define as coordenadas do ponto em que é direcionado o olhar.
ViewUpDirectionSet	Define a direção da borda superior do quadro no espaço 3D.
Renderização	
Render	Executa um ciclo completo de renderização iterando todos os objetos de cena no buffer de quadro interno para posterior exibição.
RenderBegin	Prepara um contexto gráfico para renderização de um novo quadro.
RenderEnd	Copia o quadro renderizado no buffer interno e atualiza a imagem no gráfico, se necessário.
Obtenção de recursos	
DXContext	Retorna o identificador para o contexto gráfico.
DXDispatcher	Obtém o identificador do gerenciador de recursos.
InputScene	Retorna um ponteiro para o buffer de parâmetros de cena.

© 2000-2024, MetaQuotes Ltd.

AmbientColorGet

Obtém cor e intensidade de iluminação ambiente não direcional.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // cor e intensidade da iluminação indireta
   );
```

Parâmetros

&ambient_color

[out] Cor da iluminação não direcional.

Valor retornado

Não

Observação

A intensidade é armazenada no canal alfa da estrutura DXColor.

© 2000-2024, MetaQuotes Ltd.

AmbientColorSet

Define cor e intensidade de iluminação ambiente não direcional.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // cor e intensidade da iluminação indireta
   );
```

Parâmetros

&ambient_color

[in] Cor da iluminação não direcional.

Valor retornado

Não

Observação

A intensidade é definida no canal alfa da estrutura DXColor.

© 2000-2024, MetaQuotes Ltd.

Attach

Obtém - a partir do objeto OBJ_BITMAP_LABEL - o recurso gráfico e vincula a ele uma instância da classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identificador de gráf
   const string       objname,                           // nome do objeto
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de processament
   )
```

Cria um recurso gráfico para o objeto OBJ_BITMAP_LABEL e vincula a ele uma instância da classe CCanvas.

```
bool  Attach(
   const long         chart_id,                          // identificador de gráf
   const string       objname,                           // nome do objeto
   const int          width,                             // largura da imagem em p
   const int          height,                            // altura da imagem em p
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // método de processament
   )
```

Parâmetros

chart_id

[in] Identificador do gráfico.

objname

[in] Título (nome) do objeto gráfico.

width

[in] Largura do quadro no recurso.

height

[in] Altura do quadro.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de processamento de cores. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Observação

true, se for bem-sucedido; false, se não for possível adicionar um objeto gráfico.

© 2000-2024, MetaQuotes Ltd.

Create

Cria um recurso gráfico para renderizar uma cena 3D sem vinculação ao objeto do gráfico.

```
virtual bool  Create(
   const string       name,                                // nome do objeto gráfico
   const int          width,                               // largura
   const int          height,                              // altura
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA     // formato de cor
   );
```

Parâmetros

nome

[in] Nome do objeto gráfico.

width

[in] Largura do quadro.

height

[in] Altura do quadro.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Método de processamento de cores. Saiba mais sobre os métodos de processamento de cor na descrição da função ResourceCreate().

Observação

true, se o recurso foi criado com sucesso; caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

Destroy

Destrói um recurso gráfico e libera um contexto gráfico 3D.

```
virtual void  Destroy()
```

Valor retornado

Não

Observação

Se um recurso gráfico for criado com vinculação ao objeto de gráfico, do gráfico será excluído o objeto.

© 2000-2024, MetaQuotes Ltd.

DXContext

Retorna o identificador para o contexto gráfico.

```
int  DXContext()
```

Valor retornado

Identificador de contexto gráfico.

© 2000-2024, MetaQuotes Ltd.

DXDispatcher

Obtém o identificador do gerenciador de recursos.

```
CDXDispatcher*  DXDispatcher()
```

Valor retornado

Identificador do gerenciador de recursos.

© 2000-2024, MetaQuotes Ltd.

InputScene

Retorna um ponteiro para o buffer de parâmetros de cena.

```
CDXInput*  InputScene()
```

Valor retornado

Ponteiro para um buffer de parâmetros de cena.

© 2000-2024, MetaQuotes Ltd.

LightColorGet

Obtém cor e intensidade de uma fonte de luz direcional.

```
void  LightColorGet(
   DXColor  &light_color      // cor e intensidade de iluminação direcional
   );
```

Parâmetros

&light_color

[out] Cor e intensidade da iluminação direcional.

Valor retornado

Não.

Observação

A intensidade é armazenada no canal alfa da estrutura DXColor.

© 2000-2024, MetaQuotes Ltd.

LightColorSet

Define cor e intensidade da fonte de luz direcional.

```
void  LightColorSet(
   const DXColor  &light_color      // cor e intensidade da iluminação direcional
   );
```

Parâmetros

&light_color

[in] Cor e intensidade da iluminação direcional

Valor retornado

Não.

Observação

A intensidade é definida no canal alfa da estrutura DXColor.

© 2000-2024, MetaQuotes Ltd.

LightDirectionGet

Obtém a direção de uma fonte de luz direcional.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // vetor de direção
   );
```

Parâmetros

&light_direction

[out] Vetor de direção

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

LightDirectionSet

Especifica a direção da fonte de luz direcional.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // vetor de direção
   );
```

Parâmetros

&light_direction

[in] Vetor de direção.

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

ObjectAdd

Adiciona um objeto à cena 3D para posterior renderização.

```
bool  ObjectAdd(
   CDXObject  *object      // ponteiro para o objeto
   );
```

Parâmetros

object

[in] Ponteiro para uma instância de uma classe que é descendente da classe abstrata CDXObject.

Valor retornado

true, se for bem-sucedido; false, se não for possível adicionar um objeto gráfico 3D.

© 2000-2024, MetaQuotes Ltd.

ProjectionMatrixGet

Obtém a matriz de projeção de una cena 3D em um quadro 2D.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // matriz de projeção
   );
```

Parâmetros

&projection_matrix

[out] Matriz de projeção.

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

ProjectionMatrixSet

Calcula e define a matriz de projeção de uma coordenada 3D em um quadro 2D.

```
void  ProjectionMatrixSet(
   float  fov,              // ângulo de visão (field of view)
   float  aspect_ratio,     // proporção do quadro
   float  z_near,           //
   float  z_far             //
   );
```

Parâmetros

fov

[in] Largura do campo de visão em radianos para criar uma projeção da cena.

aspect_ratio

[in] Proporção de quadro 2D.

z_near

[in] Distância até o plano de recorte próximo.

z_far

[in] Distância até o plano de recorte distante.

Valor retornado

Não.

Observação

Em um quadro 2D, serão exibidas projeções apenas dos objetos 3D que ficam dentro do campo de visão especificado e estão localizados entre os planos de recorte próximo e distante.

© 2000-2024, MetaQuotes Ltd.

Render

Executa um ciclo completo de renderização iterando todos os objetos de cena no buffer de quadro interno para posterior exibição.

```
bool  Render(
   uint  flags,                  // combinação de sinalizadores
   uint  background_color=0      // cor de fundo
   );
```

Parâmetros

flags

[in] Combinação de sinalizadores que define o modo de renderização. Valores possíveis:
DX_CLEAR_COLOR - limpar o buffer de imagem com cor *background_color*.
DX_CLEAR_DEPTH - limpar o buffer de profundidade.

background_color=0

[in] Cor de fundo da cena 3D.

Valor retornado

true, se for bem-sucedido; false, se não conseguir renderizar.

Observação

A chamada de Render() não leva à atualização da cena no gráfico, em vez disso, só atualiza o buffer interno da imagem. Para desenhar o quadro atualizado, é necessário chamar explicitamente o método Update().

Render() contém as chamadas RenderBegin e RenderEnd().

© 2000-2024, MetaQuotes Ltd.

RenderBegin

Prepara um contexto gráfico para renderização de um novo quadro.

```
virtual bool  RenderBegin(
   uint  flags,                  // combinação de sinalizadores
   uint  background_color=0      // cor de fundo
   );
```

Parâmetros

flags

[in] Combinação de sinalizadores que define o modo de renderização. Valores possíveis:
DX_CLEAR_COLOR - limpar o buffer de imagem com cor *background_color*.
DX_CLEAR_DEPTH - limpar o buffer de profundidade.

background_color=0

[in] Cor de fundo da cena 3D.

Valor retornado

true, se for bem-sucedido; false, se não for possível atualizar os parâmetros de entrada do sombreador.

© 2000-2024, MetaQuotes Ltd.

RenderEnd

Copia o quadro renderizado no buffer interno e atualiza a imagem no gráfico, se necessário.

```
virtual bool  RenderEnd(
   bool  redraw=false      // sinalizador de atualização
   );
```

Parâmetros

redraw=false

[in] Sinalizador para redesenhar o gráfico.

Valor retornado

true, se for bem-sucedido; caso contrário, false.

© 2000-2024, MetaQuotes Ltd.

ViewMatrixGet

Retorna uma matriz de vista de uma cena 3D.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // matriz de visualização
   );
```

Parâmetros

&view_matrix

[out] Matriz de visualização que define a posição e a direção da câmera no espaço 3D.

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

ViewMatrixSet

Define a matriz de vista de uma cena 3D.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // matriz de visualização
   );
```

Parâmetros

&view_matrix

[in] Matriz de visualização que define a posição e a direção da câmera no espaço 3D.

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

ViewPositionSet

Define a posição do ponto de vista em uma cena 3D.

```
void  ViewPositionSet(
   const DXVector3  &position      // posição do ponto de vista
   );
```

Parâmetros

&posiiton

[in] Posição do ponto de vista na cena 3D.

Valor retornado

Não.

Observação

A definição da posição do ponto de vista usando ViewPositionSet() altera a matriz de visualização obtida em ViewMatrixGet().

© 2000-2024, MetaQuotes Ltd.

ViewRotationSet

Define o ponto de enfoque para uma cena 3D.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // vetor de ângulos de rotação
   );
```

Parâmetros

&rotation

[in] Vetor que define os ângulos de Euler para calcular a direção de olhada em uma cena 3D.

Valor retornado

Não.

Observação

A definição da direção do olhar com ViewRotationSet() altera a matriz da visualização obtida em ViewMatrixGet().

© 2000-2024, MetaQuotes Ltd.

ViewTargetSet

Define as coordenadas do ponto em que é direcionado o olhar.

```
void  ViewTargetSet(
   const DXVector3  &target      // coordenadas alvo
   );
```

Parâmetros

&target

[in] Coordenadas do ponto em que é direcionado o olhar.

Valor retornado

Não.

Observação

Usado para fixar o olhar em um ponto da cena ao mover o ponto de vista.

A definição da nova coordenada alvo com ViewRotationSet() altera a matriz da visualização obtida em ViewMatrixGet().

ViewTargetSet() é usado em conjunto com ViewUpDirectionSet() para determinar a direção do olhar.

© 2000-2024, MetaQuotes Ltd.

ViewUpDirectionSet

Define a direção da borda superior do quadro no espaço 3D.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // direção da parte superior
   );
```

Parâmetros

&up_direction

[in] Direção da parte superior do quadro no espaço 3D.

Valor retornado

Não.

Observação

A definição da nova direção com ViewUpDirectionSet() altera a matriz de visualização obtida em ViewMatrixGet().

ViewUpDirectionSet() é usado em conjunto com ViewTargetSet() para determinar a direção do olhar.

© 2000-2024, MetaQuotes Ltd.

CChart

A classe CChart é utilizada para simplificar o acesso as propriedades do objeto gráfico "Chart".

Descrição

A classe CChart fornece acesso para as propriedades do objeto "Chart".

Declaração

```
class CChart : public CObject
```

Título

```
#include <Charts\Chart.mqh>
```

Hierarquia de herança

CObject
  CChart

Métodos de classe

Acesso para os dados protegidos	
ChartID	Identificador do gráfico
Propriedades Gerais	
Mode	Obtém/Configura o valor da propriedade "Mode" (barra, candle ou linha)
Foreground	Obtém/Configura o valor da propriedade "Foreground"
Shift	Obtém/Configura o valor da da propriedade "Shift"
ShiftSize	Obtém/Configura o valor da propriedade "ShiftSize" (em porcentagem)
AutoScroll	Obtém/Configura o valor da propriedade "AutoScroll"
Scale	Obtém/Configura o valor da propriedade "Scale"
ScaleFix	Obtém/Configura o valor da propriedade "ScaleFix" (escala do gráfico fixo ou não)
ScaleFix_11	Obtém/Configura o valor da propriedade "ScaleFix_11"(escala da carta é de 1:1, ou não)
FixedMax	Obtém/Configura o valor da propriedade "FixedMax" (preço máximo fixo)
FixedMin	Obtém/Configura o valor da propriedade "FixedMin" (preço mínimo fixo)

© 2000-2024, MetaQuotes Ltd.

Acesso para os dados protegidos	
ScalePPB	Obtém/Configura o valor da propriedade "ScalePPB" (escala será "ponto por barra" ou não)
PointsPerBar	Obtém/Configura o valor da propriedade "PointsPerBar" (em pontos por barra)
Mostrar Propriedades	
ShowOHLC	Obtém/Configura o valor da propriedade "ShowOHLC"
ShowLineBid	Obtém/Configura o valor da propriedade "ShowLineBid"
ShowLineAsk	Obtém/Configura o valor da propriedade "ShowLineAsk"
ShowLastLine	Obtém/Configura o valor da propriedade "ShowLastLine"
ShowPeriodSep	Obtém/Configura o valor da propriedade "ShowPeriodSep" (mostra períodos separados)
ShowGrid	Obtém/Configura o valor da propriedade "ShowGrid"
ShowVolumes	Obtém/Configura o valor da propriedade "ColorVolumes" (cor de volumes e níveis de posições abertas)
ShowObjectDescr	Obtém/Configura o valor da propriedade "ShowObjectDescr" (mostra a descrição para objetos gráficos)
ShowDateScale	Configura o valor da propriedade "ShowDateScale" (data de escala do gráfico)
ShowPriceScale	Configura o valor da propriedade "ShowDateScale" (preço da escala do gráfico)
Propriedade de cores	
ColorBackground	Obtém/Configura o valor da propriedade "ColorBackground" (cor de fundo do gráfico)
ColorForeground	Obtém/Configura o valor da propriedade "ColorForeground" (cor dos eixos, escala e strings OHLC do gráfico)
ColorGrid	Obtém/Configura o valor da propriedade "ColorGrid" (cor da grade)
ColorBarUp	Obtém/Configura o valor da propriedade "ColorBarUp" (cor de barras de alta, sua sombra e o contorno do corpo do candle)
ColorBarDown	Obtém/Configura o valor da propriedade "ColorBarDown" (cor das barras de baixa, sua sombra e o contorno do corpo do candle)
ColorCandleBull	Obtém/Configura o valor da Propriedade "ColorCandleBull" (cor do corpo do candle de alta)

© 2000-2024, MetaQuotes Ltd.

Acesso para os dados protegidos	
ColorCandleBear	Obtém/Configura o valor da Propriedade "ColorCandleBear" (cor do corpo do candle de baixa)
ColorChartLine	Obtém/Configura o valor da propriedade "ColorChartLine" (cor do gráfico de linhas e candles Doji)
ColorVolumes	Obtém/Configura o valor da propriedade "ColorVolumes" (cor de volumes e níveis de posições abertas)
ColorLineBid	Obtém/Configura o valor da propriedade "ColorLineBid" (cor da linha Bid)
ColorLineAsk	Obtém/Configura o valor da propriedade "ColorLineAsk" (cor da linha Ask)
ColorLineLast	Obtém/Configura o valor da propriedade "ColorLineLast" (cor da linha do último preço de oferta)
ColorStopLevels	Obtém/Configura o valor da propriedade "ColorStopLevels" (cor do SL e níveis TP)
Leia apenas as propriedades	
VisibleBars	Obtém o número total de barras visíveis do gráfico
WindowsTotal	Obtém o número total de janelas de gráfico, incluindo as sub-janelas de gráfico dos indicadores
WindowIsVisible	Obtém a visibilidade do flag da sub-janela de gráfico específica
WindowHandle	Obtém o manipulador de janela do gráfico (HWND)
FirstVisibleBar	Obtém o número da primeira barra visível do gráfico
WidthInBars	Obtém a largura da janela em barras.
WidthInPixels	Obtém a largura da sub-janela em pixels.
HeightInPixels	Obtém a altura da sub-janela em pixels.
PriceMin	Obtém preço mínimo da sub-janela específica
PriceMax	Obtém preço máximo da sub-janela específica
Propriedades	
Attach	Atribui o quadro atual para a instância de classe
FirstChart	Atribui o primeiro gráfico do terminal do cliente para a instância de classe
NextChart	Atribui o gráfico seguinte do terminal do cliente para a instância de classe
Open	Abre o gráfico com parâmetros específicos e atribuí à instância de classe

© 2000-2024, MetaQuotes Ltd.

Acesso para os dados protegidos	
Detach	Destaca o gráfico da instância de classe
Close	Fecha o gráfico, atribuído à instância de classe
BringToTop	Exibe o gráfico no topo de outros gráficos
EventObjectCreate	Define um flag para enviar notificações de um evento de nova criação do objeto no gráfico
EventObjectDelete	Define um flag para enviar notificações de um evento de exclusão de objeto no gráfico
Indicadores	
IndicatorAdd	Adiciona um indicador com manipulador específico para uma janela de determinado gráfico
IndicatorDelete	Remove um indicador com nome específico da sub-janela de determinado gráfico
IndicatorsTotal	Retorna o número de todos os indicadores aplicados a sub-janela de determinado gráfico
IndicatorName	Retorna o nome abreviado do indicador no gráfico de determinada sub-janela de gráfico
Navegação	
Navigate	Navega no gráfico
Acesso para MQL5 API	
Symbol	Obtém o símbolo do gráfico
Period	Obtém o período do gráfico
Redraw	Redesenha o gráfico, atribuído à instância de classe
GetInteger	A função retorna o valor da propriedade do objeto correspondente
SetInteger	Configura um novo valor para a propriedade do tipo integer
GetDouble	A função retorna o valor da propriedade do objeto correspondente
SetDouble	Define um novo valor para a propriedade do tipo double
GetString	A função retorna o valor da propriedade do objeto correspondente
SetString	Define um novo valor para a propriedade do tipo string
SetSymbolPeriod	Altera símbolo e período do gráfico, atribuído à instância de classe
ApplyTemplate	Aplica-se modelo específico para o gráfico

© 2000-2024, MetaQuotes Ltd.

Acesso para os dados protegidos	
ScreenShot	Cria uma imagem do gráfico específico e salva em arquivo .gif
WindowOnDropped	Obtém número de sub-janela do gráfico correspondente ao ponto de entrada do objeto (expert ou script)
PriceOnDropped	Obtém as coordenadas de preço correspondente ao ponto de entrada do objeto (expert ou script)
TimeOnDropped	Obtém as coordenadas de tempo correspondente ao ponto de entrada do objeto (expert ou script)
XOnDropped	Obtém as coordenadas X correspondente ao ponto de entrada do objeto (expert ou script)
YOnDropped	Obtém as coordenadas Y correspondente ao ponto de entrada do objeto (expert ou script)
Entrada/saída	
virtual Save	Salva parâmetros do objeto em arquivo
virtual Load	Carrega parâmetros do objeto em arquivo
virtual Type	Obtém objeto gráfico do tipo identifier

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

© 2000-2024, MetaQuotes Ltd.

ChartID

Retorna o identificador do gráfico.

```
long  ChartID() const
```

Valor de retorno

Identificador de Gráfico, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna -1.

© 2000-2024, MetaQuotes Ltd.

Mode (Método Get)

Obtém o valor da propriedade "Mode" (barra, candle ou linha)

```
ENUM_CHART_MODE  Mode() const
```

Valor de retorno

Valor da propriedade "Mode" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna WRONG_VALUE.

Mode (Método Set)

Define um novo valor à propriedade "Mode" (barra, candle ou linha).

```
bool  Mode(
   ENUM_CHART_MODE  mode      // new chart mode
   )
```

Parâmetros

mode

[In] O modo gráfico (candle, barra ou linha) de enumeração ENUM_CHART_MODE.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

Foreground (Método Get)

Obtém o valor da propriedade "Foreground"(Primeiro Plano).

```
bool  Foreground() const
```

Valor de retorno

Valor da propriedade "Foreground" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Foreground (Método Set)

Define um novo valor para a propriedade "Foreground".

```
bool  Foreground(
   bool  foreground      // new flag value
   )
```

Parâmetros

foreground

[in] O novo valor da propriedade "Primeiro plano".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Shift (Método Get)

Obtém o valor da propriedade "Shift"

```
bool  Shift() const
```

Valor de retorno

Valor da propriedade "Shift" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

Shift (Método Set)

Define um novo valor para a propriedade "Shift".

```
bool  Shift(
   bool  shift      // new flag value
   )
```

Parâmetros

shift

[in] O novo valor da propriedade "Shift".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShiftSize (Método Get)

Obtém o valor da propriedade "ShiftSize" (em porcentagem).

```
double  ShiftSize() const
```

Valor de retorno

Valor da propriedade "ShiftSize" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna EMPTY_VALUE.

ShiftSize (Método Set)

Define um novo valor para a propriedade "Shift" (em porcentagem).

```
bool  ShiftSize(
   double  shift_size      // new property value
   )
```

Parâmetros

shift_size

[in] O novo valor da propriedade "ShiftSize" (em porcentagem).

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

AutoScroll (Método Get)

Obtém o valor da propriedade "AutoScroll".

```
bool  AutoScroll() const
```

Valor de retorno

Valor da propriedade "AutoScroll" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

AutoScroll (Método Set)

Configura um novo valor para a propriedade "AutoScroll".

```
bool  AutoScroll(
   bool  autoscroll      // new flag value
   )
```

Parâmetros

autoscroll

[in] O novo valor para a propriedade "Autoscroll".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

Scale (Método Get)

Obtém o valor da propriedade "Scale".

```
int  Scale() const
```

Valor de retorno

Valor da propriedade "Scale" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna 0.

Scale (Método Set)

Define um novo valor para a propriedade "Scale".

```
bool  Scale(
   int  scale      // new value
   )
```

Parâmetros

scale

[in] O novo valor para a propriedade "Scale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ScaleFix (Método Get)

Obtém o valor da propriedade "ScaleFix" (escala fixa do gráfico ou não).

```
bool  ScaleFix() const
```

Valor de retorno

Valor da propriedade "ScaleFix" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

ScaleFix (Método Set)

Define um novo valor para a propriedade "ScaleFix".

```
bool  ScaleFix(
   bool  scale_fix      // new value
   )
```

Parâmetros

scale_fix

[in] O novo valor para a propriedade "ScaleFix".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ScaleFix_11 (Método Get)

Obtém o valor da propriedade "ScaleFix_11" (escala da carta é de 1: 1, ou não).

```
bool  ScaleFix_11() const
```

Valor de retorno

Valor da propriedade do objeto "ScaleFix_11", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

ScaleFix_11 (Método Set)

Define um novo valor para a propriedade "ScaleFix_11".

```
bool  ScaleFix_11(
   string  scale_11      // new value
   )
```

Parâmetros

scale_11

[in] O novo valor para a propriedade "ScaleFix_11".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

FixedMax (Método Get)

Obtém o valor da propriedade "FixedMax" (preço máximo fixo)

```
double  FixedMax() const
```

Valor de retorno

Valor da propriedade "FixedMax" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna EMPTY_VALUE.

FixedMax (Método Set)

Define um novo valor para a propriedade "FixedMax".

```
bool  FixedMax(
   double  max      // new fixed maximum
   )
```

Parâmetros

max

[in] O novo valor para a propriedade "FixedMax".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

FixedMin (Método Get)

Obtém o valor da propriedade "FixedMin" (preço mínimo fixo).

```
double  FixedMin() const
```

Valor de retorno

Valor da propriedade "FixedMin" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna EMPTY_VALUE.

FixedMin (Método Set)

Define um novo valor para a propriedade "FixedMin".

```
bool  FixedMax(
   double  min      // new fixed minimum
   )
```

Parâmetros

max

[in] O novo valor para a propriedade "FixedMin".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

PointsPerBar (Método Get)

Obtém o valor da propriedade "PointsPerBar" (em pontos por barra).

```
double  PointsPerBar() const
```

Valor de retorno

Valor da propriedade do objeto "PointsPerBar", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna EMPTY_VALUE.

PointsPerBar (Método Set)

Define um novo valor para a propriedade "PointsPerBar".

```
bool  PointsPerBar(
   double  ppb      // new scale (in points per bar)
   )
```

Parâmetros

ppb

[in] O novo valor para a escala (em pontos por barra).

Valor de retorno

verdadeiro se obteve êxito, falso se a escala não mudar.

© 2000-2024, MetaQuotes Ltd.

ScalePPB (Método Get)

Obtém o valor da propriedade "ScalePPB" (escala é "ponto por barra" ou não).

```
bool  ScalePPB() const
```

Valor de retorno

Valor da propriedade do objeto "ScalePPB", atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

ScalePPB (Método Set)

Define um novo valor para a propriedade "ScalePPB".

```
bool  ScalePPB(
   bool  scale_ppb      // new flag value
   )
```

Parâmetros

scale_ppb

[in] O novo valor para a propriedade "ScalePPB".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowOHLC (Método Get)

Obtém o valor da propriedade "ShowOHLC".

```
bool  ShowOHLC() const
```

Valor de retorno

Valor da propriedade "ShowOHLC" do objeto, atribuído à instância de classe. Se não houver nenhum objeto atribuído, ele retorna falso.

ShowOHLC (Método Set)

Define um novo valor para a propriedade "ShowOHLC".

```
bool  ShowOHLC(
   bool  show      // new value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowOHLC".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowLineBid (Método Get)

Obtém o valor da propriedade "ShowLineBid".

```
bool  ShowLineBid() const
```

Valor de retorno

Valor da propriedade "ShowLineBid" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowLineBid (Método Set)

Define um novo valor para a propriedade "ShowLineBid".

```
bool  ShowLineBid(
   bool  show      // new value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowLineBid".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowLineAsk (Método Get)

Obtém o valor da propriedade "ShowLineAsk".

```
bool  ShowLineAsk() const
```

Valor de retorno

Valor da propriedade "ShowLineAsk" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowLineAsk (Método Set)

Define um novo valor para a propriedade "ShowLineAsk".

```
bool  ShowLineAsk(
   bool  show      // new value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowLineAsk".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowLastLine (Método Get)

Obtém o valor da propriedade "ShowLastLine".

```
bool  ShowLastLine() const
```

Valor de retorno

Valor da propriedade "ShowLastLine" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowLastLine (Método Set)

Define um novo valor para a propriedade "ShowLastLine".

```
bool  ShowLastLine(
   bool  show      // new flag value
   )
```

Parâmetros

show

[in] Novo valor da propriedade"ShowLastLine".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowPeriodSep (Método Get)

Obtém o valor da propriedade "ShowPeriodSep" (mostra os separadores de período).

```
bool  ShowPeriodSep() const
```

Valor de retorno

Valor da propriedade do gráfico "ShowPeriodSep", atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowPeriodSep (Método Set)

Define um novo valor à propriedade "ShowPeriodSep".

```
bool  ShowPeriodSep(
   bool  show      // new value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowPeriodSep".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowGrid (Método Get)

Obtém o valor da propriedade "ShowGrid".

```
bool  ShowGrid() const
```

Valor de retorno

Valor da propriedade "ShowGrid" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowGrid (Método Set)

Define um novo valor para a propriedade "ShowGrid".

```
bool  ShowGrid(
   bool  show      // new value
   )
```

Parâmetros

show

[in] Novo valor da propriedade "ShowGrid".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowVolumes (Método Get)

Obtém o valor da propriedade "ShowVolumes".

```
bool  ShowVolumes() const
```

Valor de retorno

Valor da propriedade "ShowVolumes" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowVolumes (Método Set)

Define um novo valor à propriedade "ShowVolumes".

```
bool  ShowVolumes(
   bool  show      // new value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowVolumes".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowObjectDescr (Método Get)

Obtém o valor da propriedade"ShowObjectDescr" (mostra a descrição dos objetos gráficos).

```
bool  ShowObjectDescr() const
```

Valor de retorno

Valor da propriedade "ShowObjectDescr" do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna falso.

ShowObjectDescr (Método Set)

Define um novo valor para a propriedade "ShowObjectDescr".

```
bool  ShowObjectDescr(
   bool  show      // New value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowObjectDescr".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowDateScale

Define um novo valor para a propriedade "ShowDateScale".

```
bool  ShowDateScale(
   bool  show      // New value
   )
```

Parâmetros

show

[in] Novo valor da propriedade "ShowDateScale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ShowPriceScale

Define um novo valor para a propriedade "ShowPriceScale".

```
bool  ShowPriceScale(
   bool  show      // New value
   )
```

Parâmetros

show

[in] O novo valor da propriedade "ShowPriceScale".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ColorBackground (Método Get)

Obtém o valor da propriedade "ColorBackground" (cor de fundo do gráfico).

```
color  ColorBackground() const
```

Valor de retorno

Valor da propriedade "ColorBackground" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorBackground (Método Set)

Configura um novo valor para a propriedade "ColorBackground".

```
bool  ColorBackground(
   color  new_color      // new background color
   )
```

Parâmetros

new_color

[in] Nova cor de segundo plano (fundo).

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorForeground (Método Get)

Obtém o valor da propriedade "ColorForeground" (cor dos eixos, escala e strings OHLC do gráfico)

```
color  ColorForeground() const
```

Valor de retorno

Valor da propriedade "ColorForeground" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorForeground (Método Set)

Define um novo valor para a propriedade "ColorForeground" (para eixos, escala e a string OHLC).

```
bool  ColorForeground(
   color  new_color      // New color
   )
```

Parâmetros

new_color

[in] Nova cor para os eixos, a escala e a string OHLC.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorGrid (Método Get)

Obtém o valor da propriedade "ColorGrid" (cor da grade)

```
color  ColorGrid() const
```

Valor de retorno

Valor da propriedade "ColorBarDown" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorGrid (Método Set)

Define um novo valor para a propriedade "ColorGrid".

```
bool  ColorGrid(
   color  new_color      // new grid color
   )
```

Parâmetros

new_color

[in] Nova cor da grade.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorBarUp (Método Get)

Obtém o valor da propriedade "ColorBarUp" (cor para barras de alta, sua sombra e contorno do corpo do candle)

```
color  ColorBarUp() const
```

Valor de retorno

Valor da propriedade "ColorBarUp" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorBarUp (Método Set)

Define um novo valor para a propriedade "ColorBarUp".

```
bool  ColorBarUp(
   color  new_color      // new color for bull bars
   )
```

Parâmetros

new_color

[in] Nova cor para barras de alta.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorBarDown (Método Get)

Obtém o valor da propriedade "ColorBarDown" (cor para barras de baixa, sua sombra e contorno do corpo do candle).

```
color  ColorBarDown() const
```

Valor de retorno

Valor da propriedade "ColorBarDown" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorBarDown (Método Set)

Configura um novo valor para a propriedade "ColorBarDown".

```
bool  ColorBarDown(
   color  new_color      // new color for bear bars
   )
```

Parâmetros

new_color

[in] Nova cor para barras de baixa.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorCandleBull (Método Get)

Obtém o valor da Propriedade "ColorCandleBull" (cor corpo do candle de alta)

```
color  ColorCandleBull() const
```

Valor de retorno

Valor da propriedade "ColorCandleBull" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorCandleBull (Método Set)

Define um novo valor para a propriedade "ColorBarBull".

```
bool  ColorCandleBull(
   color  new_color      // new color for bull candle body
   )
```

Parâmetros

new_color

[in] Nova cor do corpo do candle de alta.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorCandleBear (Get Method)

Obtém o valor da Propriedade "ColorCandleBear" (cor e corpo do candle de baixa)

```
color  ColorCandleBear() const
```

Valor de retorno

Valor da propriedade "ColorCandleBear" do gráfico, atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorCandleBear (Set Method)

Sets new value for "ColorCandleBear" property.

```
bool  ColorCandleBear(
   color  new_color      // new color for bear candle body
   )
```

Parâmetros

new_color

[in] Nova cor do corpo do candle de baixa.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorChartLine (Método Get)

Obtém o valor da propriedade "ColorChartLine" (cor de gráfico de linhas e candles Doji)

```
color  ColorChartLine() const
```

Valor de retorno

Valor da Propriedade do gráfico "ColorChartLine", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorChartLine (Método Set)

Define um novo valor para a propriedade "ColorChartLine".

```
bool  ColorChartLine(
   color  new_color      // new color of the chart lines
   )
```

Parâmetros

new_color

[in] Nova cor das linhas de gráfico (candles Doji).

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorVolumes (Método Get)

Obtém o valor da propriedade "ColorVolumes" (cor de volumes e níveis das posições abertas).

```
color  ColorVolumes() const
```

Valor de retorno

Valor da Propriedade do gráfico "ColorVolumes", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorVolumes (Método Set)

Define um novo valor para a propriedade "ColorVolumes".

```
bool  ColorVolumes(
   color  new_color      // new color of the volumes (open position levels)
   )
```

Parâmetros

new_color

[in] Nova cor dos volumes (níveis posição aberta).

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorLineBid (Método Get)

Obtém o valor da propriedade "ColorLineBid" (cor da linha Bid)

```
color  ColorLineBid() const
```

Valor de retorno

Valor da propriedade do gráfico "ColorChartLine", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorLineBid (Método Set)

Define um novo valor para a propriedade "ColorLineBid".

```
bool  ColorLineBid(
   color  new_color      // new color for Bid line
   )
```

Parâmetros

new_color

[in] Nova cor para linha Bid.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorLineAsk (Método Get)

Obtém o valor da propriedade "ColorLineAsk" (cor da linha Ask)

```
color  ColorLineAsk() const
```

Valor de retorno

Valor da Propriedade do gráfico "ColorChartLine", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorLineAsk (Método Set)

Define um novo valor para a propriedade "ColorBackground".

```
bool  ColorLineAsk(
   color  new_color      // new color for Ask line
   )
```

Parâmetros

new_color

[in] Nova cor para linha Ask.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorLineLast (Método Get)

Obtém o valor da propriedade "ColorLineLast" (cor da linha de preço da última oferta)

```
color  ColorLineLast() const
```

Valor de retorno

Valor da propriedade do gráfico "ColorLineLast", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorLineLast (Método Set)

Define um novo valor para a propriedade "ColorLineLast".

```
bool  ColorLineLast(
   color  new_color      // new color of the last deal price line
   )
```

Parâmetros

new_color

[in] Nova cor da linha de preço da última oferta.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

ColorStopLevels (Método Get)

Obtém o valor da propriedade "ColorStopLevels" (cor do SL e níveis TP)

```
color  ColorStopLevels() const
```

Valor de retorno

Valor da propriedade do gráfico "ColorStopLevels", atribuído à instância de classe. Se não houver um gráfico atribuído, ele retorna CLR_NONE.

ColorStopLevels (Método Set)

Define um novo valor para a propriedade "ColorStopLevels".

```
bool  ColorStopLevels(
   color  new_color      // new color of the SL and TP price levels
   )
```

Parâmetros

new_color

[in] Nova cor dos níveis de preços do Stop Loss e do Take Profit.

Valor de retorno

verdadeiro se obteve êxito, falso se a cor não mudar.

© 2000-2024, MetaQuotes Ltd.

VisibleBars

Obtém o número total das barras visíveis do gráfico.

```
int  VisibleBars() const
```

Valor de retorno

Obtém o número total de barras visíveis do gráfico, atribuídos à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

WindowsTotal

Obtém o número total de janelas do gráfico, incluindo as sub-janelas de indicador do gráfico.

```
int  WindowsTotal() const
```

Valor de retorno

Número total de janelas, incluindo as sub-janelas de indicador do gráfico, atribuídos à ocorrência de classe. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

WindowIsVisible

Obtém a visibilidade do flag da sub-janela de determinado gráfico.

```
bool  WindowIsVisible(
   int  num      // subwindow number
   ) const
```

Parâmetros

num

[in] Número de sub-janela (0 significa a janela base).

Valor de retorno

Retorna a visibilidade do flag na subjanela do gráfico especificado, atribuído à instância de gráfico. Se não houver nenhum gráfico atribuído, ele retorna falso.

© 2000-2024, MetaQuotes Ltd.

WindowHandle

Obtém o manipulador da janela do gráfico (HWND).

```
int  WindowHandle() const
```

Valor de retorno

Manipulador da janela do gráfico, atribuído à instância de gráfico. Se não houver um gráfico atribuído, ele retorna INVALID_HANDLE.

© 2000-2024, MetaQuotes Ltd.

FirstVisibleBar

Obtém o número da primeira barra visível do gráfico.

```
int  FirstVisibleBar() const
```

Valor de retorno

Número da primeira barra visível do gráfico, atribuído à instância de gráfico. Se não houver nenhum gráfico atribuído, ele retorna -1.

© 2000-2024, MetaQuotes Ltd.

WidthInBars

Obtém a largura da janela em barras.

```
int  WidthInBars() const
```

Valor de retorno

Largura da janela em barras do gráfico, atribuído à instância de gráfico. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

WidthInPixels

Obtém a largura da sub-janela em pixels.

```
int  WidthInPixels() const
```

Valor de retorno

Largura da sub-janela em pixels no gráfico, atribuído à instância de gráfico. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

HeightInPixels

Obtém a altura da sub-janela em pixels.

```
int  HeightInPixels(
   int  num      // subwindow number
   ) const
```

Parâmetros

num

[in] Número de sub-janela (0 significa a janela base).

Valor de retorno

Altura da subjanela em pixels no gráfico, atribuído à instância de gráfico. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

PriceMin

Obtém preço mínimo da sub-janela especificada.

```
double  PriceMin(
   int  num      // subwindow number
   ) const
```

Parâmetros

num

[in] Número de sub-janela (0 significa a janela base).

Valor de retorno

Valor do preço mínimo do gráfico, atribuído à instância de classe. Se não é o gráfico atribuído, ele retorna EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

PriceMax

Obtém preço máximo da sub-janela especificada.

```
double  PriceMax(
   int  num      // subwindow number
   ) const
```

Parâmetros

num

[in] Número de sub-janela (0 significa a janela base).

Valor de retorno

Valor do preço máximo do gráfico, atribuído à instância de classe. Se não é o gráfico atribuído, ele retorna EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

Attach

Atribui ao gráfico à instância de classe.

```
void  Attach()
```

Attach

Atribuiao gráfico específico à instância de classe.

```
void  Attach(
   long  chart       // Chart identifier
   )
```

Parâmetros

chart

[in] Identificador do gráfico para atribuir.

© 2000-2024, MetaQuotes Ltd.

FirstChart

Atribui o primeiro gráfico do terminal do cliente para a instância de classe.

```
void  FirstChart()
```

© 2000-2024, MetaQuotes Ltd.

NextChart

Atribui o quadro seguinte do terminal do cliente à instância de classe.

```
void  NextChart()
```

© 2000-2024, MetaQuotes Ltd.

Open

Abre um gráfico com parâmetros especificados e atribuí à instância de classe.

```
long  Open(
   const string     symbol_name,      // Symbol name
   ENUM_TIMEFRAMES  timeframe         // Period
   )
```

Parâmetros

symbol_name

[in] Nome do símbolo. NULL significa o símbolo do gráfico corrente (para o expert anexado).

timeframe

[in] Timeframe do gráfico (enumeradorENUM_TIMEFRAMES). 0 significa o timeframe corrente.

Valor de retorno

Identificador de gráfico.

© 2000-2024, MetaQuotes Ltd.

Detach

Destaca o gráfico a partir da instância de classe

```
void  Detach()
```

© 2000-2024, MetaQuotes Ltd.

Close

Fecha o gráfico, atribuído à instância de classe

```
void  Close()
```

© 2000-2024, MetaQuotes Ltd.

BringToTop

Exibe o gráfico no topo de outros gráficos.

```
bool  BringToTop() const
```

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

EventObjectCreate

Define um flag para enviar notificações de um evento da criação de novo objeto para todos os programas MQL5 num gráfico.

```
bool  EventObjectCreate(
   bool  flag      // flag
   )
```

Parâmetros

flag

[in] O novo valor da flag.

Valor de retorno

verdadeiro - se bem sucedida, falsa - se o flag não for alterado.

© 2000-2024, MetaQuotes Ltd.

EventObjectDelete

Define um flag para enviar notificações de um evento do objeto de exclusão para todos os programas MQL5 num gráfico.

```
bool  EventObjectDelete(
   bool  flag      // flag
   )
```

Parâmetros

flag

[in] O novo valor da flag.

Valor de retorno

verdadeiro - se bem sucedida, falsa - se o flag não for alterado.

© 2000-2024, MetaQuotes Ltd.

IndicatorAdd

Adiciona um indicador com o manipulador especificado a uma janela de determinado gráfico.

```
bool  IndicatorAdd(
   int   sub_win         // number of the sub-window
   int   handle          // handle of the indicator
   );
```

Parâmetros

sub_win

[in] O número de sub-janelas do gráfico. 0 (zero) significa a principal janela do gráfico. Se o número de uma janela que não existe for especificado, uma nova janela será criada.

handle

[in] O manipulador do indicador.

Valor do Retorno

A função retorna verdadeira em caso de sucesso, caso contrário ela retorna falso. A fim de obter informações sobre o erro, chamar a função GetLastError().

Veja também

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2024, MetaQuotes Ltd.

IndicatorDelete

Remove um indicador com um nome especificado a partir da janela de determinado gráfico.

```
bool  IndicatorDelete(
   int            sub_win       // number of the subwindow
   const string   name          // short name of the indicator
   );
```

Parâmetros

sub_win

[in] Número de sub-janela do gráfico. 0 (zero) indica a sub-janela do gráfico principal.

const name

[in] O nome abreviado do indicador que é definido na propriedade INDICATOR_SHORTNAME com a função IndicatorSetString() . Para obter o nome abreviado de um indicador use a função IndicatorName() .

Valor do Retorno

Retorna verdadeiro em caso de sucesso deletando o indicador. Do contrário ele retorna falso. Para obter detalhes do erro use a função GetLastError().

Observação

Se houver dois indicadores com nomes abreviados iguais na sub-janela do gráfico, o primeiro da linha será deletado.

Se outros indicadores neste gráfico são baseados nos valores do indicador que está sendo deletado, estes indicadores também serão deletados.

Não confundir o nome abreviado do indicador e o nome do arquivo que é especificado ao criar um indicador usando as funções iCustom() e IndicatorCreate(). Se o nome abreviado de um indicador não é definido de forma explícita, o nome do arquivo contendo o código-fonte do indicador será especificado durante a compilação.

Deletando um indicador de um gráfico não significa que seu cálculo será excluído da memória do terminal. Para liberar o manuseio do indicador usar a função IndicatorRelease() .

Nome abreviado do indicador deve ser formado corretamente. A propriedade INDICATOR_SHORTNAME será escrita usando a função IndicatorSetString() . Recomenda-se que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque o indicador a ser excluído do quadro pela função IndicatorDelete() é identificado pelo nome abreviado.

Veja também

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

IndicatorsTotal

Retorna o número de todos indicadores utilizados para a janela de determinado gráfico.

```
int  IndicatorsTotal(
   int   sub_win     // number of the subwindow
   );
```

Parâmetros

sub_win

[in] Número de sub-janela do gráfico. 0 (zero) indica a sub-janela do gráfico principal.

Valor do Retorno

O número de indicadores na janela de determinado gráfico. Para obter detalhes do erro use a função GetLastError().

Observação

A função permite uma busca através de todos os indicadores anexados ao gráfico. O número de todas as janelas do gráfico pode ser obtido a partir da propriedade CHART_WINDOWS_TOTAL usando a função GetInteger ().

Veja também

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

IndicatorName

Retorna o nome abreviado do indicador pelo índice na lista de indicadores sobre a janela de determinado gráfico.

```
string  IndicatorName(
   int   sub_win      // number of the subwindow
   int   index        // index of the indicator in the list of indicators added to the
   );
```

Parâmetros

sub_win

[in] Número de sub-janela do gráfico. 0 (zero) indica a sub-janela do gráfico principal.

index

[in] índice do indicador na lista de indicadores. A numeração de indicadores começa com zero, ou seja, o primeiro indicador da lista tem o índice 0. Para obter o número de indicadores na lista usar a função IndicatorsTotal().

Valor do Retorno

O nome abreviado do indicador que é definido na propriedade INDICATOR_SHORTNAME com a função IndicatorSetString(). Para obter detalhes do erro use a função GetLastError().

Observação

Não confundir o nome abreviado do indicador e o nome do arquivo que é especificado ao criar um indicador usando as funções iCustom() e IndicatorCreate(). Se o nome abreviado de um indicador não é definido de forma explícita, o nome do arquivo contendo o código-fonte do indicador será especificado durante a compilação.

Nome abreviado do indicador deve ser formado corretamente. A propriedade INDICATOR_SHORTNAME será escrita usando a função IndicatorSetString() . Recomenda-se que o nome abreviado deve conter valores de todos os parâmetros de entrada do indicador, porque o indicador a ser excluído do quadro pela função IndicatorDelete() é identificado pelo nome abreviado.

Veja também

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2024, MetaQuotes Ltd.

Navigate

Navegando no gráfico

```
bool  Navigate(
   ENUM_CHART_POSITION  position,      // Position
   int                  shift=0        // Shift
   )
```

Parâmetros

position

[in] Valor do enumerador ENUM_CHART_POSITION.

shift=0

[in] Número de barras a deslocar.

Valor de retorno

verdadeiro se obteve êxito, falso se não navegar no gráfico .

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém símbolo do gráfico.

```
string  Symbol() const
```

Valor de retorno

Símbolo do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

Period

Obtém o período do gráfico.

```
ENUM_TIMEFRAMES  Period() const
```

Valor de retorno

Período do gráfico, atribuído à instância de classe. Se não houver nenhum gráfico atribuído, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redesenha gráfico, atribuído à instância de classe.

```
void  Redraw()
```

© 2000-2024, MetaQuotes Ltd.

GetInteger

A função retorna o valor da propriedade do objeto correspondente. A propriedade do objeto devem ser do tipo integer. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // property identifier
   int                          sub_window=0      // subwindow number
   ) const
```

2. Se for bem sucedido, coloca o valor da propriedade para a variável específica do tipo integer, passados como referência no último parâmetro.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,         // property identifier
   int                          sub_window,      // subwindow number
   long&                        value            // here we get the property value
   ) const
```

Parâmetros

prop_id

[in] Identificador de propriedade (enumerador ENUM_CHART_PROPERTY_INTEGER).

sub_window

[in] Número da sub-janela do gráfico.

value

[In] Variável do tipo integer que recebeu o valor da propriedade solicitada.

Valor do Retorno

Valor de propriedade do gráfico, atribuído à instância de classe. Se não houver qualquer gráfico atribuído, ele retorna -1.

Para a segunda variante a função retornará verdadeira, se essa propriedade é mantida e o valor foi colocado na variável "value", caso contrário, retorna falsa. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetInteger

Define um novo valor para a propriedade do tipo integer.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // property identifier
   long                         value         // new value
   )
```

Parâmetros

prop_id

[in] Identificador de propriedade (enumerador ENUM_CHART_PROPERTY_INTEGER).

value

[in] O novo valor da propriedade.

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade do tipo integer não mudou.

© 2000-2024, MetaQuotes Ltd.

GetDouble

A função retorna o valor da propriedade do objeto correspondente. A propriedade do objeto devem ser do tipo double. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // property identifier
   int                         sub_window=0     // subwindow number
   ) const
```

2. Se for bem sucedida, coloca o valor da propriedade para a variável específica do tipo double, passada por referência como último parâmetro.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // property identifier
   int                         sub_window,     // subwindow number
   double&                     value           // here we get the property value
   ) const
```

Parâmetros

prop_id

[in] Identificador de propriedade(enumerador ENUM_CHART_PROPERTY_DOUBLE).

sub_window

[in] Número da sub-janela do gráfico.

value

[In] Variável do tipo double que recebeu o valor da propriedade solicitada.

Valor do Retorno

Valor de propriedade do gráfico, atribuído à instância de classe. Se não houver qualquer gráfico atribuído, ele retorna EMPTY_VALUE.

Para a segunda variante a função retornará verdadeira, se essa propriedade é mantida e o valor foi colocado na variável "value", caso contrário, retorna falsa. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetDouble

Define um novo valor para a propriedade do tipo double.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // property identifier
   double                      value         // new value
   )
```

Parâmetros

prop_id

[in] Identificador da propriedade do enumerador (ENUM_CHART_PROPERTY_DOUBLE).

value

[in] O novo valor para a propriedade.

Valor de retorno

Verdadeiro se obteve êxito, falso se a propriedade do tipo double não mudou.

© 2000-2024, MetaQuotes Ltd.

GetString

A função retorna o valor da propriedade do objeto correspondente. A propriedade do objeto deve ser do tipo string. Existem 2 variantes da função.

1. Imediatamente retorna o valor da propriedade.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // property identifier
   ) const
```

2. Se for bem sucedido, coloca o valor da propriedade para a variável específica do tipo string, passados como referência no último parâmetro.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // property identifier
   string&                     value        // here we get the property value
   ) const
```

Parâmetros

prop_id

[in] Identificador de Propriedade (enumerador ENUM_CHART_PROPERTY_STRING).

sub_window

[in] Número da sub-janela do gráfico.

value

[In] Variável do tipo string que recebeu o valor da propriedade solicitada.

Valor do Retorno

Valor de propriedade do gráfico, atribuído à instância de classe. Se não houver qualquer gráfico atribuído, ele retorna "".

Para a segunda variante a função retornará verdadeira, se essa propriedade é mantida e o valor foi colocado na variável "value", caso contrário, retorna falsa. Para ler mais sobre o erro, chamar GetLastError().

© 2000-2024, MetaQuotes Ltd.

SetString

Define um novo valor para a propriedade do tipo string.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // property identifier
   string                      value         // new property value
   )
```

Parâmetros

prop_id

[in] Identificador de propriedade (enumerador ENUM_CHART_PROPERTY_STRING).

value

[in] O novo valor para a propriedade.

Valor de retorno

Verdadeiro se obteve êxito, falso se propriedade do tipo string não mudou.

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Altera símbolo e período do gráfico, atribuído à instância de classe.

```
bool  SetSymbolPeriod(
   const string     symbol_name,      // Symbol
   ENUM_TIMEFRAMES  timeframe         // Period
   )
```

Parâmetros

symbol_name

[in] Novo nome do símbolo. NULL significa o símbolo do gráfico corrente (para o expert anexado).

timeframe

[in] Novo timeframe de gráfico (enumerador ENUM_TIMEFRAMES). 0 significa o timeframe corrente.

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

ApplyTemplate

Aplica um template específico ao gráfico.

```
bool  ApplyTemplate(
   const string  filename      // template file name
   )
```

Parâmetros

filename

[in] nome de arquivo do template.

Valor de retorno

Verdadeiro se obteve êxito, falso se o template não for aplicado.

© 2000-2024, MetaQuotes Ltd.

ScreenShot

Cria a captura de tela do gráfico especificado e salva como arquivo ".gif".

```
bool  ScreenShot(
   string           filename,                  // File name
   int              width,                     // Width
   int              height,                    // Height
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Align type
   ) const
```

Parâmetros

filename

[in] O nome do arquivo para captura de tela.

width

[in] Captura de tela com largura em pixels.

height

[in] Captura de tela com altura em pixels.

align_mode=ALIGN_RIGHT

[in] Modo alinhar, se a captura de tela é estreita.

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

WindowOnDropped

Obtém número gráfico da sub-janela correspondente ao ponto de entrada do objeto (expert ou script).

```
int  WindowOnDropped() const
```

Valor de retorno

Número do gráfico da sub-janela do ponto de entrada do objeto. 0 significa a janela principal do gráfico.

© 2000-2024, MetaQuotes Ltd.

PriceOnDropped

Obtém coordenada do preço correspondente ao ponto de entrada do objeto (expert ou script).

```
double  PriceOnDropped() const
```

Valor de retorno

Coordenar preço do ponto de entrada do objeto.

© 2000-2024, MetaQuotes Ltd.

TimeOnDropped

Obtém as coordenadas de tempo correspondente ao ponto de entrada do objeto (expert ou script).

```
datetime  TimeOnDropped() const
```

Valor de retorno

Coordenar o tempo do ponto de entrada do objeto.

© 2000-2024, MetaQuotes Ltd.

XOnDropped

Obtém as coordenadas X correspondente ao ponto de entrada do objeto (expert ou script).

```
int  XOnDropped() const
```

Valor de retorno

Coordenada X do ponto de entrada do objeto.

© 2000-2024, MetaQuotes Ltd.

YOnDropped

Obtém as coordenadas Y correspondente ao (expert ou script)

```
int  YOnDropped() const
```

Valor de retorno

Coordenada Y do ponto de entrada do objeto.

© 2000-2024, MetaQuotes Ltd.

Save

Salva parâmetros do objeto para arquivo.

```
virtual bool  Save(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen(...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega parâmetros do objeto a partir do arquivo.

```
virtual bool  Load(
   int  file_handle      // File handle
   )
```

Parâmetros

file_handle

[in] Manipulador para o arquivo binário já aberto pela função FileOpen(...).

Valor de retorno

verdadeiro se obteve êxito, falso se foi um erro.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o identificador do tipo de objeto gráfico

```
virtual int  Type() const
```

Valor de retorno

Identificador do tipo de objeto (0x1111 para CChart).

© 2000-2024, MetaQuotes Ltd.

Gráficos científicos

Biblioteca gráfica que contém as classes e funções globais para plotagem rápida de gráficos personalizados no gráfico. A biblioteca fornece uma solução cômoda e pronta para a construção de eixos, curvas, bem como métodos de acesso rápido para alterar as propriedades gerais de um gráfico personalizado.

Para começar a trabalhar com a biblioteca, basta ler o artigo Visualize isto! Biblioteca gráfica em linguagem MQL5 como equivalente a plot de R.

A biblioteca gráfica está localizada no diretório de trabalho do terminal na pasta Include\Graphics.

Classe	Descrição
CAxis	Classe para trabalhar com os eixos coordenados
ColorGenerator	Classe que especifica o esquema de cores padrão
CCurve	Classe para trabalhar com curvas
CGraphic	Classe base para a criação de gráficos personalizados

© 2000-2024, MetaQuotes Ltd.

GraphPlot

Função de plotagem rápida de curvas.

Versão para plotagem de uma curva nas coordenadas Y.

```
string  GraphPlot(
   const double      &y[],                  // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Observação

As coordenadas X para dada curva serão os índices de matriz Y.

Versão para plotagem de uma curva nas coordenadas X e Y.

```
string  GraphPlot(
   const double      &x[],                  // coordenadas X
   const double      &y[],                  // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Versão para plotagem de dois curvas nas coordenadas X e Y.

```
string  GraphPlot(
   const double      &x1[],                 // coordenadas X
   const double      &y1[],                 // coordenadas Y
   const double      &x2[],                 // coordenadas X
   const double      &y2[],                 // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

Versão para plotagem de três curvas nas coordenadas X e Y.

```
string  GraphPlot(
   const double      &x1[],                 // coordenadas X
   const double      &y1[],                 // coordenadas Y
   const double      &x2[],                 // coordenadas X
   const double      &y2[],                 // coordenadas Y
   const double      &x3[],                 // coordenadas X
   const double      &y3[],                 // coordenadas Y
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // tipo de curva
   )
```

© 2000-2024, MetaQuotes Ltd.

Versão para plotagem da curva nos pontos CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points[],            // coordenadas da curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versão para plotagem de dois curvas nos pontos CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordenadas da curva
   const CPoint2D   &points2[],           // coordenadas da curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versão para plotagem de três curvas nos pontos CPoint2D

```
string  GraphPlot(
   const CPoint2D   &points1[],           // coordenadas da curva
   const CPoint2D   &points2[],           // coordenadas da curva
   const CPoint2D   &points3[],           // coordenadas da curva
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versão para plotagem da curva de acordo com o ponteiro para a função CurveFunction

```
string  GraphPlot(
   CurveFunction    function,             // ponteiro para a função
   const double     from,                 // valor inicial do argumento
   const double     to,                   // valor final do argumento
   const double     step,                 // incremento do argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Versão para plotagem de dois curvas de acordo com o ponteiro para a função CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // ponteiro para a função
   CurveFunction    function2,            // ponteiro para a função
   const double     from,                 // valor inicial do argumento
   const double     to,                   // valor final do argumento
   const double     step,                 // incremento do argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

© 2000-2024, MetaQuotes Ltd.

Versão para plotagem de três curvas de acordo com o ponteiro para a função CurveFunction

```
string  GraphPlot(
   CurveFunction    function1,            // ponteiro para a função
   CurveFunction    function2,            // ponteiro para a função
   CurveFunction    function3,            // ponteiro para a função
   const double     from,                 // valor inicial do argumento
   const double     to,                   // valor final do argumento
   const double     step,                 // incremento do argumento
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // tipo de curva
   )
```

Parâmetros

&x[]

[in] Координаты X.

&y[]

[in] Coordenadas Y.

&x1[]

[in] Coordenadas X para a primeira curva.

&y1[]

[in] Coordenadas X para a segunda curva.

&x2[]

[in] Coordenadas X para a segunda curva.

&y2[]

[in] Coordenadas Y para a segunda curva.

&x3[]

[in] Coordenadas X para a terceira curva.

&y3[]

[in] Coordenadas Y para a terceira curva.

&points[]

[in] Coordenadas dos pontos da curva.

&points1[]

[in] Coordenadas dos pontos da primeira curva.

&points2[]

[in] Coordenadas dos pontos da segunda curva.

&points3[]

[in] Coordenadas dos pontos da terceira curva.

© 2000-2024, MetaQuotes Ltd.

function

[in] Ponteiro para a função CurveFunction.

function1

[in] Ponteiro para a primeira função.

function2

[in] Ponteiro para a segunda função.

function3

[in] Ponteiro para a terceira função.

from

[in] Corresponde à primeira coordenada X.

to

[in] Corresponde à última coordenada X.

step

[in] Parâmetro para calcular as coordenadas de X.

type=CURVE_POINTS

[in] Tipo de curva.

Valor de retorno

Nome do recurso gráfico.

© 2000-2024, MetaQuotes Ltd.

CAxis

Classe CAxis — classe auxiliar da biblioteca gráfica para trabalhar com os eixos coordenados.

Descrição

A classe CAxis fornece armazenamento e aquisição de vários parâmetros dos eixos coordenados. Na classe, implementa-se a possibilidade dimensionar automática e dinamicamente os eixos coordenados.

Declaração

```
class CAxis
```

Cabeçalho

```
#include <Graphics\Axis.mqh>
```

Métodos de classe

Método	Descrição
AutoScale	Obter/definir sinalizador de dimensionamento automático
Min	Obter/definir o valor mínimo do eixo
Max	Obter/definir o valor máximo do eixo
Step	Retorna o valor de incremento no eixo
Name	Obter/definir o nome do eixo
Color	Obter/definir a cor do eixo
ValuesSize	Obter/definir o tamanho dos números no eixo
ValuesWidth	Obter/definir o comprimento máximo dos números no eixo
ValuesFormat	Obter/definir o formato dos números no eixo
ValuesDateTimeMode	Retorna o formato de conversão de data para uma cadeia de caracteres.
ValuesFunctionFormat	Retorna o ponteiro para a função que define o formato de saída dos valores no eixo.
ValuesFunctionFormatCBData	Retorna o ponteiro para o objeto que pode conter informações adicionais para converter os valores do eixo.
NameSize	Obter/definir o tamanho da fonte para o nome do eixo
ZeroLever	Obter/definir o valor de "alavanca zero"

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
DefaultStep	Obter/definir o valor inicial de incremento no eixo
MaxLabels	Obter/definir o número máximo de dígitos no eixo
MinGrace	Obter/definir o valor de "tolerância" para o mínimo do eixo
MaxGrace	Obter/definir o valor de "tolerância" para o máximo do eixo
SelectAxisScale	Executa o dimensionamento automático do eixo.

© 2000-2024, MetaQuotes Ltd.

AutoScale (método Get)

Ele retorna o sinalizador que marca se é necessário executar o dimensionamento automático.

```
bool  AutoScale()
```

Valor de retorno

Valor do sinalizador.

Observação

true — executar o dimensionamento automático.

false — não executar o dimensionamento automático.

AutoScale (método Set)

Ele define o sinalizador que marca se é necessário executar o dimensionamento automático.

```
void  AutoScale(
   const bool  auto      // valor do sinalizador
   )
```

Parâmetros

auto

[in]

Observação

true — executar o dimensionamento automático.

false — não executar o dimensionamento automático.

© 2000-2024, MetaQuotes Ltd.

Min (método Get)

Retorna o valor mínimo do eixo.

```
double  Min()
```

Valor de retorno

Valor mínimo do eixo.

Min (método Set)

Define o valor mínimo do eixo.

```
void  Min(
   const double  min      // valor mínimo
   )
```

Parâmetros

min

[in] Valor mínimo.

© 2000-2024, MetaQuotes Ltd.

Max (método Get)

Retorna o valor máximo do eixo.

```
double  Max()
```

Valor de retorno

Valor máximo do eixo.

Max (método Set)

Define o valor máximo do eixo no gráfico.

```
void  Max(
   const double  max      // valor máximo
   )
```

Parâmetros

max

[in] Valor máximo do eixo no gráfico.

© 2000-2024, MetaQuotes Ltd.

Step (método Get)

Retorna o valor de incremento no eixo.

```
double  Step()
```

Valor de retorno

Valor do incremento.

© 2000-2024, MetaQuotes Ltd.

Name (método Get)

Retorna o nome do eixo.

```
string  Name()
```

Valor de retorno

Nome do eixo.

Name (método Set)

Define o nome do eixo.

```
void  Name(
   const string  name      // nome do eixo
   )
```

Parâmetros

name

[in] Nome do eixo.

© 2000-2024, MetaQuotes Ltd.

Color (método Get)

Retorna a cor do eixo.

```
color  Color()
```

Valor de retorno

Cor do eixo.

Color (método Set)

Define a cor do eixo.

```
void  Color(
   const color  clr      // cor do eixo
   )
```

Parâmetros

clr

[in] cor do eixo.

© 2000-2024, MetaQuotes Ltd.

ValuesSize (método Get)

Retorna o tamanho dos números no eixo.

```
int  ValuesSize()
```

Valor de retorno

Tamanho dos número no eixo.

ValuesSize (método Set)

Define o tamanho dos números no eixo.

```
void  ValuesSize(
   const int  size      // tamanho dos número no eixo
   )
```

Parâmetros

size

[in] Tamanho dos números no eixo

© 2000-2024, MetaQuotes Ltd.

ValuesWidth (método Get)

Retorna o comprimento máximo permitido, em pixels, para exibir números nos eixos.

```
int  ValuesWidth()
```

Valor de retorno

Comprimento dos números exibidos no eixo em pixels.

Observação

Se o comprimento em pixels para o número definido for superior ao comprimento máximo permitido de exibição, esse comprimento vai ser cortado e terminado usando reticências.

ValuesWidth (método Set)

Define o comprimento máximo permitido, em pixels, para exibir números nos eixos.

```
void  ValuesWidth(
   const int  width      // comprimento máximo permitido em pixels
   )
```

Parâmetros

width

[in] Comprimento máximo permitido de exibição de números no eixo.

Observação

Se o comprimento em pixels para o número definido for superior ao comprimento máximo permitido de exibição, esse comprimento vai ser cortado e terminado usando reticências.

© 2000-2024, MetaQuotes Ltd.

ValuesFormat (método Get)

Retorna o formato para exibir números no eixo.

```
string  ValuesFormat()
```

Valor de retorno

Formato de exibição de números.

ValuesFormat (método Set)

Define o formato para exibir números no eixo.

```
void  ValuesFormat(
   const string  format      // formato para exibir números no eixo
   )
```

Parâmetros

format

[in] formato para exibir números no eixo.

© 2000-2024, MetaQuotes Ltd.

ValuesDateTimeMode (Método Get)

Retorna o formato de conversão de data para uma cadeia de caracteres.

```
int  ValuesDateTimeMode()
```

Valor de retorno

Formato de conversão de data para cadeia de caracteres.

ValuesDateTimeMode (Método Set)

Define o formato de conversão de data para uma cadeia de caracteres.

```
void  ValuesDateTimeMode(
   const int  mode      // formato de conversão de data para uma cadeia de caracteres
   )
```

Parâmetros

mode

[in] Formato de conversão.

Observação

Saiba mais sobre formatos de conversão de data para cadeia na descrição da função TimeToString().

© 2000-2024, MetaQuotes Ltd.

ValuesFunctionFormat (Método Get)

Retorna o ponteiro para a função que define o formato de saída dos valores no eixo.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Valor de retorno

Ponteiro para a função que define o formato de saída dos valores no eixo.

ValuesFunctionFormat (Método Set)

Define o ponteiro para a função que define o formato de saída dos valores no eixo.

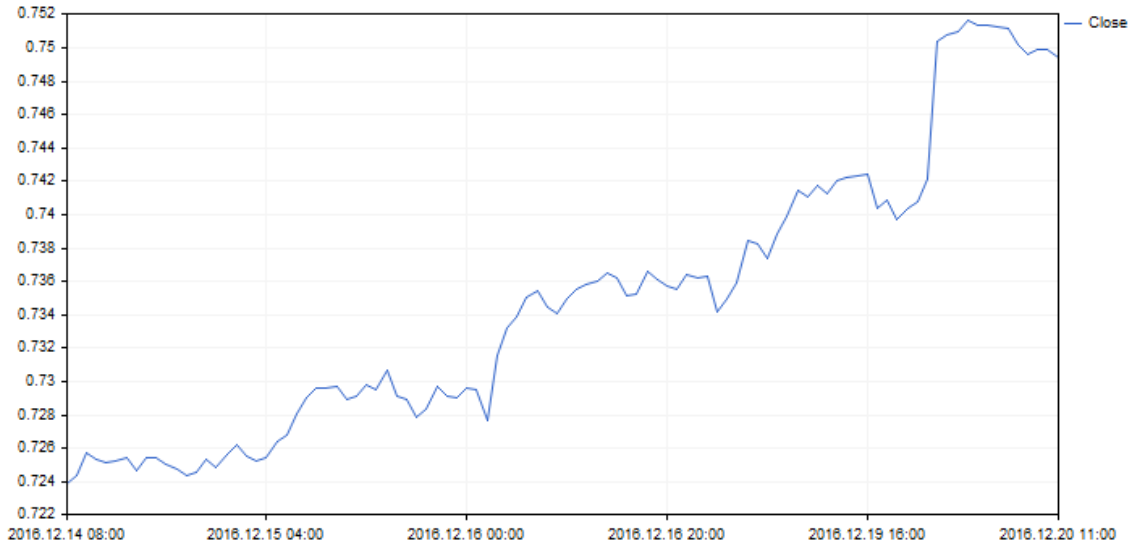
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // função para converter valores numéricos para u
   )
```

Parâmetros

func

[in] Função personalizada para converter valores numéricos para uma cadeia de caracteres.

Exemplo:



O formato de saída dos valores no eixo X foi alterado com o seguinte código:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              DateAxisGraphic.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- array for store values
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| Custom function for create values on X-axis                      |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- gets the X-axis
   CAxis *xAxis=graphic.XAxis();
//--- sets the X-axis properties
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

ValuesFunctionFormatCBData (Método Get)

Retorna o ponteiro para o objeto que pode conter informações adicionais para converter os valores do eixo.

```
void*  ValuesFunctionFormatCBData()
```

Valor de retorno

Ponteiro para o objeto que pode conter informações adicionais para converter os valores do eixo.

ValuesFunctionFormatCBData (Método Set)

Define o ponteiro para o objeto que pode conter informações adicionais para converter os valores do eixo.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // ponteiro para objeto de classe
   )
```

Parâmetros

cbdata

[in] Ponteiro para o objeto de qualquer classe que contém informações adicionais para converter os valores do eixo

© 2000-2024, MetaQuotes Ltd.

NameSize (método Get)

Retorna o tamanho da fonte para o nome do eixo.

```
int  NameSize()
```

Valor de retorno

Tamanho da fonte pelo qual é definido o nome do eixo.

NameSize (método Set)

Define o tamanho da fonte para o nome do eixo.

```
void  NameSize(
   const int  size      // tamanho da fonte para o nome do eixo
   )
```

Parâmetros

size

[in] Tamanho da fonte pelo qual é definido o nome do eixo.

© 2000-2024, MetaQuotes Ltd.

ZeroLever (método Get)

Retorna o valor de "alavanca zero."

```
double  ZeroLever()
```

Valor de retorno

"Alavanca zero."

Observação

Este valor é utilizado para determinar quando o intervalo de escala do eixo deve ser expandido para incluir o valor de zero.

ZeroLever (método Set)

Define o valor de "alavanca zero."

```
void  ZeroLever(
   const double  value      // valor de "alavanca zero"
   )
```

Parâmetros

value

[in] Valor de "alavanca zero."

Observação

Este valor é utilizado para determinar quando o intervalo de escala do eixo deve ser expandido para incluir o valor de zero.

© 2000-2024, MetaQuotes Ltd.

DefaultStep (método Get)

Retorna o valor inicial de incremento no eixo.

```
double  DefaultStep()
```

Valor de retorno

Incremento do eixo.

DefaultStep (método Set)

Define o valor inicial de incremento no eixo.

```
void  DefaultStep(
   const double  value      // incremento do eixo
   )
```

Parâmetros

value

[in] Valor inicial de incremento no eixo.

© 2000-2024, MetaQuotes Ltd.

MaxLabels (método Get)

Retorna o número máximo de dígitos permitido no eixo.

```
double  MaxLabels()
```

Valor de retorno

Número máximo de dígitos no eixo.

MaxLabels (método Set)

Define o número máximo de dígitos permitido no eixo.

```
void  MaxLabels(
   const double  value      // número máximo
   )
```

Parâmetros

value

[in] Número máximo permitido de dígitos exibidos no eixo.

© 2000-2024, MetaQuotes Ltd.

MinGrace (método Get)

Retorna o valor de "tolerância" aplicado ao mínimo do eixo.

```
double  MinGrace()
```

Valor de retorno

Valor de "tolerância" para o mínimo do eixo.

Observação

Este valor é expresso como parte do comprimento axial total. Por exemplo, se os valores do eixo se encontrarem na faixa 4.0 - 16.0, seu comprimento será igual a 12.0. Se MaxGrace for igual a 0.1, 10% do comprimento do eixo (ou 1.2) será subtraído do valor do mínimo. Como resultado, o eixo cobrirá a faixa 2.8 - 16.0.

MinGrace (método Set)

Define o valor de "tolerância" aplicado ao mínimo do eixo.

```
void  MinGrace(
   const double  value      // valor de "tolerância"
   )
```

Parâmetros

value

[in] Valor de "tolerância" aplicado ao mínimo do eixo.

Observação

Este valor é expresso como parte do comprimento axial total. Por exemplo, se os valores do eixo se encontrarem na faixa 4.0 - 16.0, seu comprimento será igual a 12.0. Se MaxGrace for igual a 0.1, 10% do comprimento do eixo (ou 1.2) será subtraído do valor do mínimo. Como resultado, o eixo cobrirá a faixa 2.8 - 16.0.

© 2000-2024, MetaQuotes Ltd.

MaxGrace (método Get)

Retorna o valor de "tolerância" aplicado ao máximo do eixo.

```
double  MaxGrace()
```

Valor de retorno

Valor de "tolerância" para o máximo do eixo

Observação

Este valor é expresso como parte do comprimento axial total. Por exemplo, se os valores do eixo se encontrarem na faixa 4.0 - 16.0, seu comprimento será igual a 12.0. Se MaxGrace for igual a 0.1, 10% do comprimento do eixo (ou 1.2) será adicionado ao valor do máximo. Como resultado, o eixo cobrirá a faixa 4.0 - 17.2.

MaxGrace (método Set)

Define o valor de "tolerância" aplicado ao máximo do eixo.

```
void  MaxGrace(
   const double  value      // valor de "tolerância"
   )
```

Parâmetros

value

[in] Valor de tolerância aplicado ao máximo do eixo.

Observação

Este valor é expresso como parte do comprimento axial total. Por exemplo, se os valores do eixo se encontrarem na faixa 4.0 - 16.0, seu comprimento será igual a 12.0. Se MaxGrace for igual a 0.1, 10% do comprimento do eixo (ou 1.2) será subtraído do valor do mínimo. Como resultado, o eixo cobrirá a faixa 2.8 - 16.0.

© 2000-2024, MetaQuotes Ltd.

SelectAxisScale

Executa o dimensionamento automático do eixo.

```
void  SelectAxisScale()
```

© 2000-2024, MetaQuotes Ltd.

CColorGenerator

Classe CColorGenerator — classe auxiliar da biblioteca gráfica para trabalhar com a paleta de cores.

Descrição

A classe CColorGenerator contém uma paleta inicial de cores utilizadas, por padrão, para as curvas (se a cor não for especificada pelo usuário).

Se todas as cores da paleta primária forem usadas, serão automaticamente geradas novas cores e será recargada a paleta de cores.

Declaração

```
class CColorGenerator
```

Cabeçalho

```
#include <Graphics\ColorGenerator.mqh>
```

Métodos de classe

Método	Descrição
Next	Retorna a próxima cor da paleta
Reset	Leva o gerador ao seu estado original

© 2000-2024, MetaQuotes Ltd.

Next

Retorna a seguinte cor da paleta.

```
uint  Next()
```

Valor de retorno

Cor.

Observação

Se todas as cores da paleta forem baralhadas, começará automaticamente a geração de novas cores que serão substituídas na paleta antiga.

© 2000-2024, MetaQuotes Ltd.

Reset

Leva o gerador ao seu estado original.

```
void  Reset()
```

© 2000-2024, MetaQuotes Ltd.

CCurve

A classe CCurve é uma classe para trabalhar com as propriedades das curvas geradas no gráfico.

Descrição

A classe CCurve fornece instalação, armazenamento e aquisição de coordenadas e vários parâmetros das curvas ao trabalhar com a classe CGraphic.

Há três modos de plotagem de curvas: pontos, linhas e barra de gráficos. Para cada modo de plotagem, na classe, são implementados parâmetros separados.

Declaração

```
class CCurve : public CObject
```

Cabeçalho

```
#include <Graphics\Curve.mqh>
```

Hierarquia de herança

CObject
  CCurve

Métodos de classe

Método	Descrição
tipo	Retorna o tipo de curva
Adicione, edite e remova suas contas de mensageiro instantâneo e VoIP. Name	Retorna o nome da curva
Color	Retorna a cor da curva
XMax	Retorna o valor máximo da função no eixo X
XMin	Retorna o valor mínimo da função no eixo X
YMax	Retorna o valor máximo da função no eixo Y
YMin	Retorna o valor mínimo da função no eixo Y
Size	Retorna o número de pontos que definem a curva
PointsSize	Obter/definir as dimensões lineares dos pontos que definem a curva
PointsFill	Obter/definir o sinalizador de preenchimento dos pontos que definem a curva
PointsColor	Obter/definir a cor de preenchimento dos pontos
GetX	Obtém na matriz os valores de X para todos os pontos da curva

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
GetY	Obtém na matriz os valores de Y de todos os pontos da curva
LinesStyle	Obter/definir o estilo de linha ao plotar uma curva usando linhas
LinesIsSmooth	Obter/definir o sinalizador de suavização ao desenhar linhas
LinesSmoothTension	Obter/definir o parâmetro de suavização da curva durante a plotagem usando linhas
LinesSmoothStep	Obter/definir o comprimento das linhas que se aproximam para suavização durante a plotagem usando linhas
LinesWidth	Obter/definir a largura das linhas ao plotar uma curva usando linhas
HistogramWidth	Obter/definir a largura da coluna ao plotar o gráfico de barras
CustomPlotCBData	Obter/definir o ponteiro para o objeto que será usado durante o modo personalizado de plotagem de curva.
CustomPlotFunction	Obter/definir o ponteiro para a função que implementa o modo personalizado de plotagem de curva.
PointsType	Obter/definir o sinalizador que indica o tipo de pontos que são usados ao plotar uma curva usando pontos.
StepsDimension	Obter/definir o valor que indica a dimensão segundo a qual é feito o passo durante a plotagem em escada de uma curva.
TrendLineCoefficients	Obter/definir os coeficientes da linha de tendência para gravá-los numa matriz.
TrendLineColor	Obter/definir a cor da linha de tendência para a curva.
TrendLineVisible	Obter/definir o sinalizador de visibilidade da linha de tendência.
Update	Atualiza as coordenadas da curva.
Visible	Obter/definir o sinalizador que indica se a função é visível no gráfico.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o tipo da curva.

```
ENUM_CURVE_TYPE  Type()
```

Valor de retorno

Tipo de curva.

© 2000-2024, MetaQuotes Ltd.

Name

Retorna o nome da curva.

```
string  Name()
```

Valor de retorno

Nome da curva.

© 2000-2024, MetaQuotes Ltd.

Color

Retorna a cor da curva.

```
uint  Color()
```

Valor de retorno

Cor da curva.

© 2000-2024, MetaQuotes Ltd.

XMax

Retorna o valor máximo para a função no eixo X (apenas números reais).

```
double  XMax()
```

Valor de retorno

Número real máximo entre todos os argumentos para dada função.

© 2000-2024, MetaQuotes Ltd.

XMin

Retorna o valor mínimo da função no eixo X (apenas valores reais).

```
double  XMin()
```

Valor de retorno

Número real mínimo entre todos os argumentos para dada função.

© 2000-2024, MetaQuotes Ltd.

YMax

Retorna o valor máximo para a função no eixo Y (apenas números reais).

```
double  YMax()
```

Valor de retorno

Valor máximo para dada função no eixo Y (apenas números reais).

© 2000-2024, MetaQuotes Ltd.

YMin

Retorna o valor mínimo para a função no eixo Y (apenas números reais).

```
double  YMin()
```

Valor de retorno

Valor mínimo para dada função no eixo Y (apenas números reais).

© 2000-2024, MetaQuotes Ltd.

Size

Retorna o número de pontos que definem dada curva.

```
int  Size()
```

Valor de retorno

Número de pontos que definem a curva.

© 2000-2024, MetaQuotes Ltd.

PointsSize (método Get)

Retorna as dimensões lineares dos pontos que definem dada curva durante sua plotagem, em pixels.

```
int  PointsSize()
```

Valor de retorno

Tamanho dos pontos que definem a curva, em pixels.

PointsSize (método Set)

Define o tamanho dos pontos que definem dada curva durante sua plotagem, em pixels.

```
void  PointsSize(
   const int  size      // tamanho dos pontos em pixels
   )
```

Parâmetros

size

[in] Tamanho dos pontos que definem esta curva, em pixels.

© 2000-2024, MetaQuotes Ltd.

PointsFill (método Get)

Retorna o sinalizador que indica se é necessário executar o preenchimento para os pontos que definem a curva durante a plotagem usando pontos.

```
bool  PointsFill ()
```

Valor de retorno

Valor do sinalizador.

Observação

true — executar o preenchimento

false — não executar o preenchimento

PointsFill (método Set)

Define o sinalizador que indica se é necessário executar o preenchimento para os pontos que definem a curva durante a plotagem usando pontos.

```
void  PointsFill(
   const bool  fill      // valor do sinalizador
   )
```

Parâmetros

fill

[in] Valor do sinalizador.

Observação

true — executar o preenchimento

false — não executar o preenchimento

© 2000-2024, MetaQuotes Ltd.

PointsColor (método Get)

Retorna a cor de preenchimento dos pontos.

```
uint  PointsColor ()
```

Valor de retorno

Cor de preenchimento dos pontos que definem dada curva.

PointsColor (método Set)

Define a cor de preenchimento dos pontos.

```
void  PointsColor(
   const uint  clr      //cor de preenchimento dos pontos
   )
```

Parâmetros

clr

[in] Cor de preenchimento dos pontos que definem dada curva.

© 2000-2024, MetaQuotes Ltd.

GetX

Obtém na matriz os valores de X para todos os pontos da curva.

```
void  GetX(
   double&  x[]      // matriz para registro dos valores X
   )
```

Parâmetros

x[]

[out] Matriz para obtenção dos valores de X para todos os pontos da curva.

Observação

Cada ponto da curva é especificado por um par de valores X e Y. Esses valores não são em coordenadas de pixels para plotagem na classe *** CGraphic.

© 2000-2024, MetaQuotes Ltd.

GetY

Obtém na matriz os valores de Y de todos os pontos da curva.

```
void  GetY(
   double&  y[]      // matriz para registro dos valores Y
   )
```

Parâmetros

y[]

[out] Matriz para obtenção dos valores de Y para todos os pontos da curva.

Observação

Cada ponto da curva é especificado por um par de valores X e Y. Esses valores não são em coordenadas de pixels para plotagem na classe *** CGraphic.

© 2000-2024, MetaQuotes Ltd.

LinesStyle (método Get)

Retorna o estilo de linha ao plotar a curva usando linhas.

```
ENUM_LINE_STYLE  LinesStyle()
```

Valor de retorno

Estilo de linha.

LinesStyle (método Set)

Define o estilo de linha ao plotar a curva usando linhas.

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // estilo de linha
   )
```

Parâmetros

style

[in] Estilo de linha.

© 2000-2024, MetaQuotes Ltd.

LinesIsSmooth (método Get)

Retorna o sinalizador que define se é necessário executar a suavização durante a plotagem da curva usando linhas.

```
bool  LinesIsSmooth()
```

Valor de retorno

Valor do sinalizador

Observação

true — executar a suavização

false — não executar a suavização

LinesIsSmooth (método Set)

Define o sinalizador que define se é necessário executar a suavização durante a plotagem da curva usando linhas.

```
void  LinesIsSmooth(
   const bool  smooth      // valor do sinalizador
   )
```

Parâmetros

smooth

[in] Valor do sinalizador

Observação

true — executar a suavização

false — não executar a suavização

© 2000-2024, MetaQuotes Ltd.

LinesSmoothTension (método Get)

Retorna o parâmetro de suavização da curva durante a plotagem usando linhas.

```
double  LinesSmoothTension()
```

Valor de retorno

O valor do parâmetro de suavização

Observação

Valor tension localizado no intervalo (0.0; 1.0].

LinesSmoothTension (método Set)

Define o parâmetro de suavização da curva durante a plotagem usando linhas.

```
void  LinesSmoothTension(
   const double  tension      // valor do parâmetro
   )
```

Parâmetros

tension

[in] Valor do parâmetro de suavização.

Observação

Valor tension localizado no intervalo (0.0; 1.0].

© 2000-2024, MetaQuotes Ltd.

LinesSmoothStep (método Get)

Retorna o comprimento das linhas que se aproximam para suavização durante a plotagem usando linhas

```
double  LinesSmoothStep()
```

Valor de retorno

Comprimento das linhas que se aproximam em pixels.

LinesSmoothStep (método Set)

Define o comprimento das linhas que se aproximam para suavização durante a plotagem usando linhas.

```
void  LinesSmoothStep(
   const double  step      // comprimento das linhas
   )
```

Parâmetros

step

[in] Comprimento das linhas que se aproximam

© 2000-2024, MetaQuotes Ltd.

LinesEndStyle (Método Set)

Retorna o sinalizador que indica o estilo de plotagem de extremidades de linhas ao desenhar uma curva usando linhas .

```
ENUM_LINE_END  LinesEndStyle()
```

Valor de retorno

Valor do sinalizador que indica o estilo de extremos de linhas ao desenhar uma curva usando linhas.

LinesEndStyle (Método Get)

Define o sinalizador que indica o estilo de plotagem de extremidades de linhas ao desenhar uma curva usando linhas

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // valor do sinalizador
   )
```

Parâmetros

end_style

[in] Valor do sinalizador que indica o estilo de extremos de linha ao desenhar uma curva usando linhas.

© 2000-2024, MetaQuotes Ltd.

LinesWidth (Método Get)

Retorna a largura das linhas ao plotar a curva usando linhas.

```
int  LinesWidth()
```

Valor de retorno

Largura das linhas.

LinesWidth (Método Set)

Define a largura das linhas ao plotar a curva usando linhas.

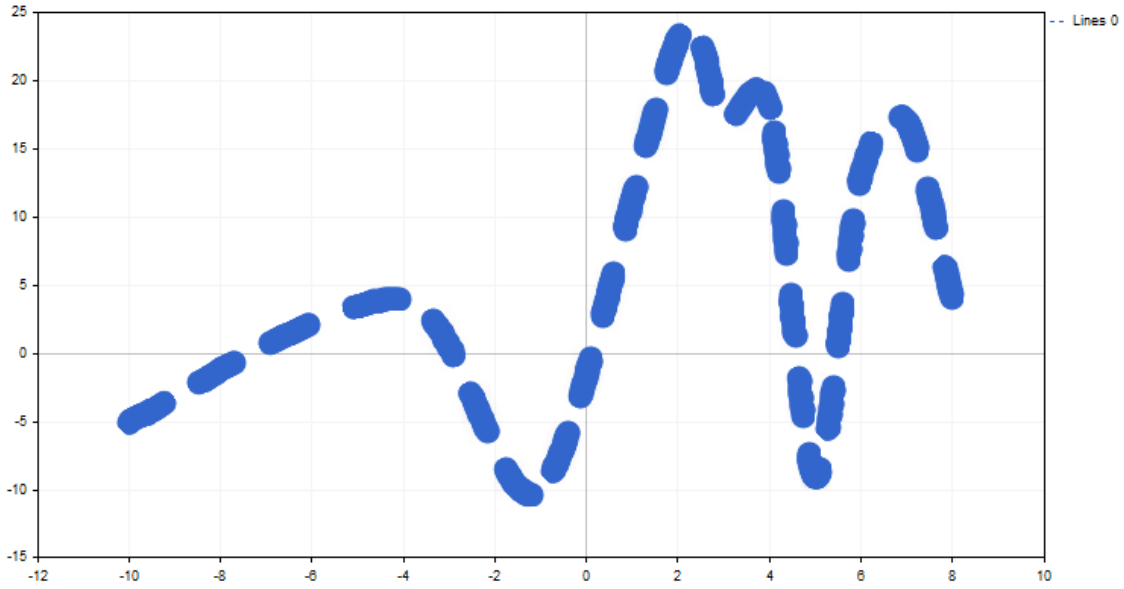
```
void  LinesWidth(
   const int  width      // largura das linhas
   )
```

Parâmetros

width

[in] Largura das linhas ao plotar a curva usando linhas.

Exemplo:



A espessura de linha é alterada usando o seguinte código:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- sets the curve properties
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

HistogramWidth (método Get)

Retorna a largura da coluna ao plotar o gráfico de barras.

```
int  HistogramWidth()
```

Valor de retorno

Largura da coluna em pixels.

HistogramWidth (método Set)

Define a largura da coluna ao plotar o gráfico de barras.

```
void  HistogramWidth(
   const int  width      // largura das colunas
   )
```

Parâmetros

width

[in] Largura das colunas em pixels.

© 2000-2024, MetaQuotes Ltd.

CustomPlotCBData (Método Get)

Retorna o ponteiro para o objeto que será usado durante o modo personalizado de plotagem de curva.

```
void*  CustomPlotCBData()
```

Valor de retorno

Ponteiro para o objeto que será usado durante o modo personalizado de plotagem de curva.

CustomPlotCBData (Método Set)

Define o ponteiro para o objeto que será usado durante o modo personalizado de plotagem de curva.

```
void  CustomPlotCBData(
   void*  cbdata      // ponteiro para o objeto
   )
```

Parâmetros

cbdata

[in] Ponteiro para o objeto que será usado durante o modo personalizado de plotagem de curva

© 2000-2024, MetaQuotes Ltd.

CustomPlotFunction (Método Get)

Retorna o ponteiro para a função que implementa o modo personalizado de plotagem de curva.

```
PlotFucntion  CustomPlotFunction()
```

Valor de retorno

Ponteiro para a função que implementa o modo personalizado de plotagem de curva.

CustomPlotFunction (Método Set)

Define o ponteiro para a função que implementa o modo personalizado de plotagem de curva.

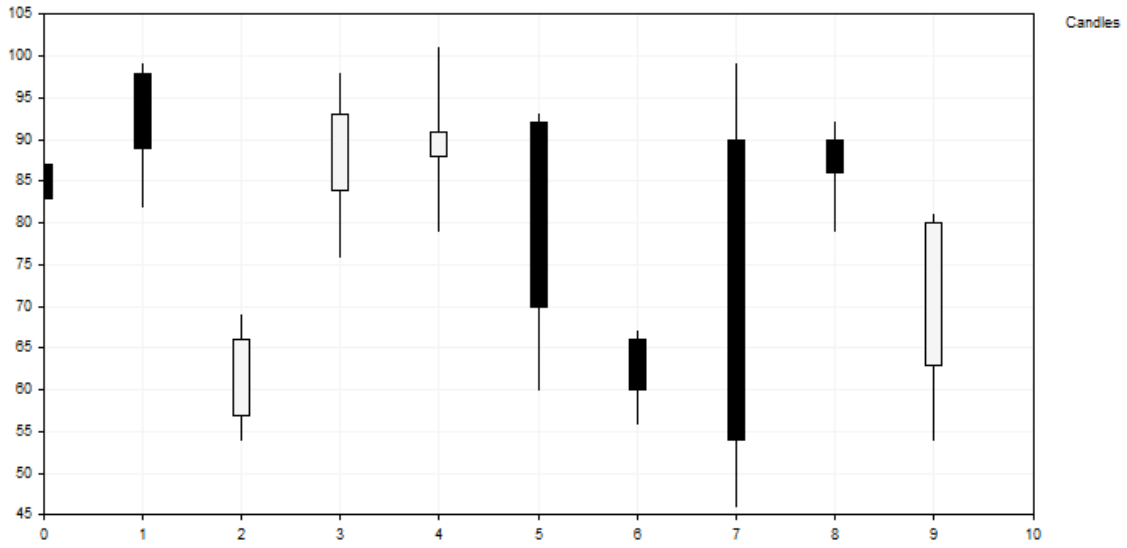
```
void  CustomPlotFunction(
   PlotFucntion  func      // ponteiro para a função
   )
```

Parâmetros

func

[in] Ponteiro para a função que implementa o modo personalizado de plotagem de curva

Exemplo:



Esta curva das barras foi plotada usando o seguinte código:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Class CCandle                                                    |
//| Usage: class to represent the candle                             |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double high,co
                                                    const int width,const uint clr_
                    ~CCandle(void);
   double            OpenValue(void)            const { return(m_open);     }
   double            CloseValue(void)           const { return(m_close);    }
   double            HigthValue(void)           const { return(m_high);     }
   double            LowValue(void)             const { return(m_low);      }
   uint              CandleColorIncrement(void) const { return(m_clr_inc);  }
   uint              CandleColorDecrement(void) const { return(m_clr_dec);  }
   int               CandleWidth(void)          const { return(m_width);    }
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double l
                                 const int width,const uint clr_inc=0x000000,const uin
                                 m_open(open),m_close(close),m_high(high),m_low(low),
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Custom method for plot candles                                   |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- check obj
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- plot candles
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

```
      if(candle==NULL)
         return;
      //--- primary calculate
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ? candle.CandleColorIncrement() :  candle.CandleColorDec
      //--- plot candle
      canvas.LineVertical(xc,high,low,0x000000);
      //--- plot candle real body
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- create values
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- calculate values
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- find max and min
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- create candle
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- sets the curve properties
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

© 2000-2024, MetaQuotes Ltd.

```
//--- sets the graphic properties
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2024, MetaQuotes Ltd.

PointsType (Método Get)

Retorna o sinalizador que indica o tipo de pontos que são usados ao plotar uma curva usando pontos.

```
ENUM_POINT_TYPE  PointsType()
```

Valor de retorno

Valor do sinalizador que indica o tipo de pontos.

PointsType (Método Set)

Define o sinalizador que indica o tipo de pontos que são usados ao plotar uma curva usando pontos.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // valor do sinalizador
   )
```

Parâmetros

type

[in] Valor do sinalizador que indica o tipo de pontos que são usados ao plotar uma curva usando pontos.

© 2000-2024, MetaQuotes Ltd.

StepsDimension (Método Get)

Retorna o valor que indica a dimensão segundo a qual é feito o passo durante a plotagem em escada de uma curva.

```
int  StepsDimension()
```

Valor de retorno

Dimensão segundo a qual é feito o passo durante a plotagem em escada de uma curva.

StepsDimension (Método Set)

Define o valor que indica a dimensão segundo a qual é feito o passo durante a plotagem em escada de uma curva.

```
void  StepsDimension(
   const int  dimension      // dimensão
   )
```

Parâmetros

dimension

[in] Dimensão (pode ser 0 ou 1).

Observação

0 — x (primeiro é traçada a linha horizontal e, depois, a vertical).

1 — y (primeiro é traçada a linha vertical e, depois, a horizontal).

© 2000-2024, MetaQuotes Ltd.

TrendLineCoefficients (Método Get)

Retorna os coeficientes da linha de tendência para gravá-los numa matriz.

```
double&  TrendLineCoefficients()
```

Valor de retorno

Coeficientes de linha de tendência.

TrendLineCoefficients (Método Set)

Obtém os coeficientes da linha de tendência para gravá-los numa matriz.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // matriz para armazenamento de coeficientes
   )
```

Parâmetros

coefficients[]

[out] Matriz para armazenamento de coeficientes.

© 2000-2024, MetaQuotes Ltd.

TrendLineColor (Método Get)

Retorna a cor da linha de tendência para a curva.

```
uint  TrendLineColor()
```

Valor de retorno

Cor da linha de tendência.

TrendLineColor (Método Set)

Define a cor da linha de tendência para a curva.

```
void  TrendLineColor(
   const uint  clr      // cor da linha de tendência
   )
```

Parâmetros

clr

[in] Cor de linha

© 2000-2024, MetaQuotes Ltd.

TrendLineVisible (Método Get)

Retorna o sinalizador que determina a visibilidade da linha de tendência.

```
bool  TrendLineVisible()
```

Valor de retorno

Valor do sinalizador que indica se a linha de tendência é visível.

TrendLineVisible (Método Set)

Define o sinalizador que determina a visibilidade da linha de tendência.

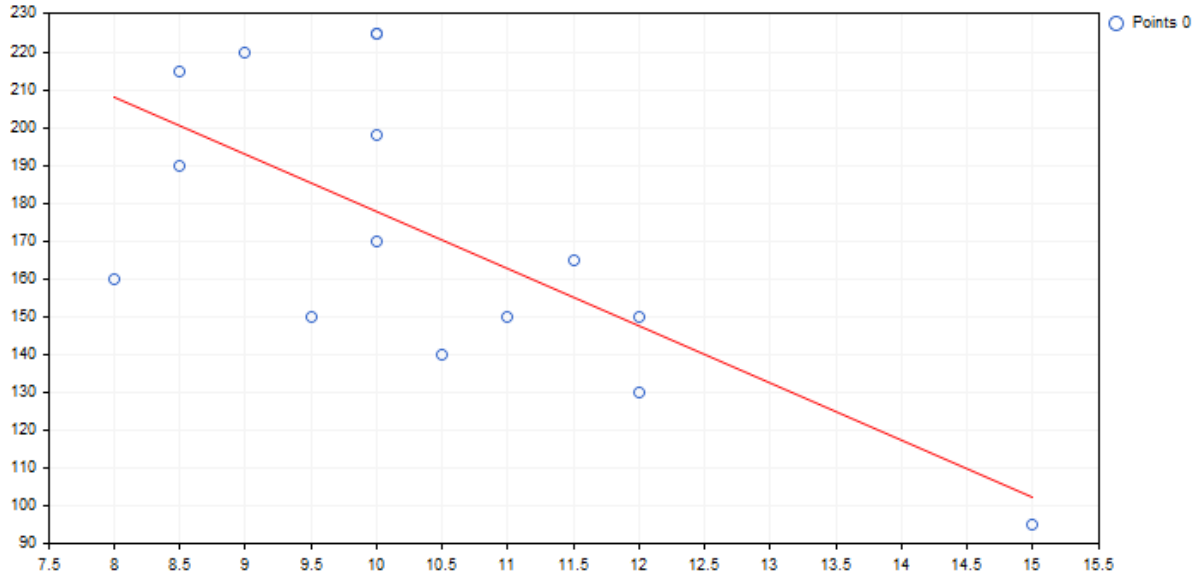
```
void  TrendLineVisible(
   const bool  visible      // valor do sinalizador
   )
```

Parâmetros

visible

[in] Valor do sinalizador que determina a visibilidade da linha de tendência.

Exemplo:



Código de plotagem da linha de tendência descrita acima e sua exibição no gráfico.

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0
//--- create graphic
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- create curve
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- sets the curve properties
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- plot
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

Update

Atualiza as coordenadas da curva.

Versão para trabalhar na coordenada Y. Neste caso, os índices da matriz transferida são as coordenada X.

```
void  Update(
   const double&  y[]   // Coordenadas Y
   )
```

Versão para trabalhar sobre o par de coordenadas X e Y.

```
void  Update(
   const double&  x[],     // Coordenadas X
   const double&  y[]      // Coordenadas Y
   )
```

Versão para trabalhar com os pontos CPoint2D.

```
void  Update(
   const CPoint2D&  points[]      // Coordenadas da curva
   )
```

Versão para trabalhar com o ponteiro para a função CurveFunction.

```
void  Update(
   CurveFunction  function,     // ponteiro para a função que descreve a curva
   const double   from,         // valor inicial do argumento da função
   const double   to,           // valor final do argumento da função
   const double   step          // incremento do argumento
   )
```

Parâmetros

x[]

[in] Coordenadas X.

y[]

[in] Coordenadas Y.

points[]

[in] Coordenadas da curva.

function

[in] Ponteiro para a função que descreve a curva

from

[in] Valor inicial do argumento da função

to

[in] Valor final do argumento da função

step

© 2000-2024, MetaQuotes Ltd.

[in] Incremento do argumento

© 2000-2024, MetaQuotes Ltd.

Visible (Método Get)

Retorna o sinalizador que indica se a função é visível no gráfico.

```
void  Visible(
   const bool  visible      //
   )
```

Valor de retorno

Valor do sinalizador que indica se a função é visível no gráfico.

Visible (Método Set)

Define o sinalizador que indica se a função é visível no gráfico.

```
void  Visible(
   const bool  visible      // valor do sinalizador
   )
```

Parâmetros

visible

[in] Valor do sinalizador que indica se a função é visível no gráfico.

© 2000-2024, MetaQuotes Ltd.

CGraphic

Classe CGraphic — classe base para criação de gráficos personalizados.

Descrição

A classe CGraphic proporciona inúmeros aspectos do trabalho com gráficos personalizados.

A classe armazena os elementos básicos do gráfico, define seus parâmetros, realiza a plotagem.

Além disso, essa classe armazena as curvas para os gráficos e oferece várias opções para exibi-los.

Declaração

```
class CGraphic
```

Cabeçalho

```
#include <Graphics\Graphic.mqh>
```

Métodos de classe

Método	Descrição
Create	Cria um recurso gráfico que está vinculado ao objeto de gráfico
Destroy	Exclui gráficos e destrói recursos gráficos
Update	É exibida na tela as mudanças realizadas
ChartObjectName	Retorna o nome do objeto vinculado ao gráfico
ResourceName	Retorna o nome do recurso gráfico
XAxis	Retorna o ponteiro para o eixo X
YAxis	Retorna o ponteiro para o eixo Y
GapSize	Obter/definir o tamanho do recuo entre os elementos do gráfico
BackgroundColor	Obter/definir a cor de fundo
BackgroundMain	Obter/definir o texto do cabeçalho do gráfico
BackgroundMainSize	Obter/definir tamanho da fonte do cabeçalho
BackgroundMainColor	Obter/definir cor do cabeçalho do gráfico
BackgroundSub	Obter/definir o texto da legenda
BackgroundSubSize	Obter/definir tamanho da fonte do cabeçalho
BackgroundSubColor	Obter/definir a cor de legenda do gráfico
GridLineColor	Obter/definir a cor das linhas de grade

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
GridBackgroundColor	Obter/definir a cor de fundo da grade
GridCircleRadius	Obter/definir o raio dos pontos nos nós da rede
GridCircleColor	Obter/definir a cor dos pontos nos nós da rede
GridHasCircle	Obter/definir o sinalizador de plotagem de pontos nos nós da rede
GridAxisLineColor	Retorna o valor de cor de eixos reais do gráfico.
HistoryNameWidth	Obter/definir o comprimento máximo permitido para exibir os nomes da curva
HistoryNameSize	Obter/definir tamanho da fonte no nome da curva
HistorySymbolSizeo	Obter/definir tamanho dos símbolos
TextAdd	Adiciona texto ao gráfico
LineAdd	Adiciona linhas ao gráfico
CurveAdd	Cria e adiciona curvas ao gráfico
CurvePlot	Desenha uma curva previamente criada de acordo com o índice
CurvePlotAll	Desenha todas as curvas criadas anteriormente
CurveGetByIndex	Obtém a curva no índice especificado
CurveGetByName	Obtém a curva de acordo com o nome especificado
CurveRemoveByIndex	Exclui a curva segundo o índice especificado.
CurveRemoveByName	Exclui a curva segundo o nome especificado.
CurvesTotal	Retorna o número de curvas para um determinado gráfico.
MarksToAxisAdd	Adiciona as marcas de escala no eixo do gráfico
MajorMarkSize	Obter/definir o tamanho de marcas da escala no eixo do gráfico
FontSet	Definir os parâmetros da fonte atual
FontGet	Obter os parâmetros da fonte atual
Attach	Obter/criar um recurso gráfico e vinculá-lo à instância da classe CGraphic
CalculateMaxMinValues	Executa cálculo (recálculo) dos valores mínimo e máximo do gráfico em ambos os eixos.
Height	Retorna a altura do gráfico em pixels.

© 2000-2024, MetaQuotes Ltd.

Método	Descrição
IndentDown	Obter/definir o recuo a partir da borda inferior do gráfico.
IndentLeft	Obter/definir o recuo a partir da borda esquerda do gráfico.
IndentRight	Obter/definir o recuo a partir da borda direita do gráfico.
IndentUp	Obter/definir o recuo a partir da borda superior do gráfico.
Redraw	Redesenha o gráfico.
ResetParameters	Restaura os parâmetros necessários para redesenhar o gráfico.
ScaleX	Dimensiona o valor no eixo X.
ScaleY	Dimensiona o valor no eixo Y.
SetDefaultParameters	Define os parâmetros do gráfico para os valores padrão.
Width	Retorna a largura do gráfico em pixels.

© 2000-2024, MetaQuotes Ltd.

Create

Cria um recurso gráfico que está vinculado ao objeto de gráfico.

```
bool  Create(
   const long    chart,       // identificador do gráfico
   const string  name,        // nome
   const int     subwin,      // número da subjanela
   const int     x1,          // coordenada x1
   const int     y1,          // coordenada y1
   const int     x2,          // coordenada x2
   const int     y2           // coordenada y1
   )
```

Parâmetros

chart

[in] Identificador do gráfico.

name

[in] Nome.

subwin

[in] Número da subjanela.

x1

[in] Coordenada X1.

y1

[in] Coordenada Y1.

x2

[in] Coordenada X2.

y2

[in] Coordenada Y2.

© 2000-2024, MetaQuotes Ltd.

Destroy

Exclui gráficos e destrói recursos gráficos.

```
void  Destroy()
```

© 2000-2024, MetaQuotes Ltd.

Update

Exibe na tela as alterações realizadas.

```
void  Update(
   const bool  redraw=true      // флаг
   )
```

Parâmetros

redraw=true

[in] Valor do sinalizador

© 2000-2024, MetaQuotes Ltd.

ChartObjectName

Recebe o nome do objeto vinculado ao gráfico.

```
string  ChartObjectName()
```

Valor de retorno

Nome do objeto vinculado ao gráfico.

© 2000-2024, MetaQuotes Ltd.

ResourceName

Recebe o nome do recurso gráfico.

```
string  ResourceName()
```

Valor de retorno

Nome do recurso gráfico.

© 2000-2024, MetaQuotes Ltd.

XAxis

Retorna o ponteiro para o eixo X.

```
CAxis  *XAxis()
```

Valor de retorno

Ponteiro para o eixo X.

© 2000-2024, MetaQuotes Ltd.

YAxis

Retorna o ponteiro para o eixo Y.

```
CAxis  *YAxis()
```

Valor de retorno

Ponteiro para o eixo Y.

© 2000-2024, MetaQuotes Ltd.

GapSize (método Get)

Retorna o tamanho do recuo entre os elementos do gráfico.

```
int  GapSize()
```

Valor de retorno

Tamanho do recuo, em pixels.

GapSize (método Set)

Define o tamanho do recuo entre os elementos do gráfico.

```
void  GapSize(
   const int  size     // tamanho do recuo
   )
```

Parâmetros

size

[in] Tamanho do recuo em pixels.

© 2000-2024, MetaQuotes Ltd.

BackgroundColor (método Get)

Retorna a cor de fundo.

```
color  BackgroundColor()
```

BackgroundColor (método Set)

Define a cor de fundo.

```
void  BackgroundColor(
   const color  clr      // cor de fundo
   )
```

Parâmetros

clr

[in] Cor de fundo.

© 2000-2024, MetaQuotes Ltd.

BackgroundMain (método Get)

Retorna o texto do cabeçalho do gráfico

```
string  BackgroundMain()
```

BackgroundMain (método Set)

Define o texto do cabeçalho do gráfico.

```
void  BackgroundMain(
   const string  main      // texto do cabeçalho
   )
```

Parâmetros

main

[in] Texto do cabeçalho do gráfico

© 2000-2024, MetaQuotes Ltd.

BackgroundMainSize (método Get)

Retorna o tamanho do cabeçalho.

```
int  BackgroundMainSize()
```

Valor de retorno

Tamanho da fonte do cabeçalho.

BackgroundMainSize (método Set)

Define o tamanho do cabeçalho.

```
void  BackgroundMainSize(
   const int  size     // tamanho do cabeçalho
   )
```

Parâmetros

size

[in] Tamanho da fonte do cabeçalho.

© 2000-2024, MetaQuotes Ltd.

BackgroundMainColor (método Get)

Retorna a cor do cabeçalho.

```
color  BackgroundMainColor()
```

Valor de retorno

Cor do cabeçalho.

BackgroundMainColor (método Set)

Define a cor do cabeçalho.

```
void  BackgroundMainColor(
   const color  clr      // cor de cabeçalho
   )
```

Parâmetros

clr

[in] Cor do cabeçalho.

© 2000-2024, MetaQuotes Ltd.

BackgroundSub (método Get)

Retorna o texto da legenda.

```
string  BackgroundSub()
```

Valor de retorno

Texto da legenda.

BackgroundSub (método Set)

Define o texto da legenda.

```
void  BackgroundSub (método Set)(
   const string  sub      // texto da legenda
   )
```

Parâmetros

sub

[in] Texto da legenda.

© 2000-2024, MetaQuotes Ltd.

BackgroundSubSize (método Get)

Retorna o tamanho de texto da legenda

```
int BackgroundSubSize()
```

BackgroundSubSize (método Get)

Define o tamanho de legenda

```
void  BackgroundSubSize(
   const int  size      // tamanho de fonte da legenda
   )
```

Parâmetros

size

[in] Tamanho de fonte da legenda

© 2000-2024, MetaQuotes Ltd.

BackgroundSubColor (método Get)

Retorna cor de legenda.

```
color  BackgroundSubColor()
```

BackgroundSubColor (método Set)

Define a cor de legenda.

```
void  BackgroundSubColor(
   const color  clr      // cor de legenda
   )
```

Parâmetros

clr

[in] Cor de legenda.

© 2000-2024, MetaQuotes Ltd.

GridLineColor (método Get)

Retorna a cor de linhas da grade

```
color  GridLineColor()
```

Valor de retorno

Cor de linhas da grade.

GridLineColor (método Set)

Define a cor de linhas da grade.

```
void  GridLineColor(
   const color  clr      // cor de linhas
   )
```

Parâmetros

clr

[in] Cor de linhas da grade.

© 2000-2024, MetaQuotes Ltd.

GridBackgroundColor (método Get)

Retorna cor de fundo da grade

```
color  GridBackgroundColor()
```

Valor de retorno

Cor de fundo da grade

GridBackgroundColor (método Set)

Define a cor de fundo da grade

```
void  GridBackgroundColor(
   const color  clr      // cor de fundo da grade
   )
```

Parâmetros

clr

[in] Cor de fundo da grade

© 2000-2024, MetaQuotes Ltd.

GridCircleRadius (método Get)

Retorna o raio dos pontos nos nós da rede.

```
int  GridCircleRadius()
```

Valor de retorno

Raio dos pontos em pixels.

GridCircleRadius (método Set)

Define o raio dos pontos nos nós da rede

```
void  GridCircleRadius(
   const int  r      // raio
   )
```

Parâmetros

r

[in] Raio dos pontos em pixels.

© 2000-2024, MetaQuotes Ltd.

GridCircleColor (método Get)

Retorna a cor dos pontos nos nós da rede.

```
color  GridCircleColor()
```

Valor de retorno

Cor dos pontos.

GridCircleColor (método Set)

Define a cor dos pontos nos nós da rede.

```
void  GridCircleColor(
   const color  clr      // cor dos pontos
   )
```

Parâmetros

clr

[in] Cor dos pontos.

© 2000-2024, MetaQuotes Ltd.

GridHasCircle (método Get)

Define o sinalizador que indica se é necessário plotar os pontos nos nós da rede.

```
bool  GridHasCircle()
```

Valor de retorno

Valor do sinalizador.

Observação

true — plotar pontos

false — não plotar pontos

GridHasCircle (método Set)

Define o sinalizador que indica se é necessário plotar os pontos nos nós da rede.

```
void  GridHasCircle(
   const bool  has
   )
```

Parâmetros

has

[in] Valor do sinalizador.

Observação

true — plotar pontos

false — não plotar pontos

© 2000-2024, MetaQuotes Ltd.

HistoryNameWidth (método Get)

Retorna o comprimento máximo permitido para exibir os nomes da curva.

```
int  HistoryNameWidth()
```

Valor de retorno

Comprimento máximo de pixels.

Observação

Se o nome da curva for superior ao comprimento máximo permitido, esse nome será cortado e terminado usando reticências.

HistoryNameWidth (método Set)

Define o comprimento máximo permitido para exibir os nomes da curva.

```
void  HistoryNameWidth(
   const int  width      // comprimento máximo
   )
```

Parâmetros

width

[In] Comprimento máximo em pixels.

Observação

Se o nome da curva for superior ao comprimento máximo permitido, esse nome será cortado e terminado usando reticências.

© 2000-2024, MetaQuotes Ltd.

HistoryNameSize (método Get)

Retorna o tamanho da fonte para o nome da curva.

```
int  HistoryNameSize()
```

Valor de retorno

Tamanho da fonte para o nome da curva.

HistoryNameSize (método Set)

Define o tamanho da fonte para o nome da curva.

```
void  HistoryNameSize (método Set)(
   const int  size      // tamanho de fonte do nome
   )
```

Parâmetros

size

[in] Tamanho de fonte do nome.

© 2000-2024, MetaQuotes Ltd.

HistorySymbolSize (método Get)

Retorna o tamanho dos símbolos para o gráfico

```
int  HistorySymbolSize()
```

Valor de retorno

Tamanho dos símbolos

HistorySymbolSize (método Set)

Define o tamanho dos símbolos para o gráfico

```
void  HistorySymbolSize(
   const int  size      // tamanho do símbolo
   )
```

Parâmetros

size

[in] Tamanho dos símbolos.

© 2000-2024, MetaQuotes Ltd.

TextAdd

Adiciona texto ao gráfico.

Versão para trabalhar com o par de coordenadas X e Y

```
void  TextAdd(
   const int     x,              // coordenada X
   const int     y,              // coordenada Y
   const string  text,           // texto
   const uint    clr,            // cor
   const uint    alignment=0     // alinhamento
   )
```

Versão para CPoint

```
void  TextAdd(
   const CPoint  &point,         // coordenada dos pontos
   const string  text,           // texto
   const uint    clr,            // cor
   const uint    alignment=0     // alinhamento
   )
```

Parâmetros

x

[in] Coordenada X.

y

[in] Coordenada Y.

&point

[in] Coordenada do ponto.

text

[in] Texto.

clr

[in] Cor.

alignment=0

[in] Alinhamento.

© 2000-2024, MetaQuotes Ltd.

LineAdd

Adiciona uma linha no gráfico.

Versão para trabalhar com o par de coordenadas X e Y

```
void  LineAdd(
   const int   x1,         // coordenada x1
   const int   y1,         // coordenada y1
   const int   x2,         // coordenada x2
   const int   y2,         // coordenada y2
   const uint  clr,        // cor
   const uint  style       // estilo
   )
```

Versão para CPoint

```
void  LineAdd2(
   const CPoint  &point1,      // coordenada do primeiro ponto
   const CPoint  &point2,      // coordenada do segundo ponto
   const uint    clr,          // cor
   const uint    style         // estilo
   )
```

Parâmetros

x1

[in] Coordenada X1.

y1

[in] Coordenada Y1.

x2

[in] Coordenada X2.

y2

[in] Coordenada Y2.

&point1

[in] Coordenada do primeiro ponto.

&point2

[in] Coordenada do segundo ponto.

clr

[in] Cor.

style

[in] Estilo.

© 2000-2024, MetaQuotes Ltd.

CurveAdd

Cria e adiciona uma nova curva no gráfico.

Versão para trabalhar na coordenada Y (a cor da curva é definido automaticamente)

```
CCurve* CurveAdd(
   const double      &y[],          // coordenadas Y
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Observação

As coordenadas X para dada curva serão os índices de matriz Y.

Versão para trabalhar nas coordenadas Y e Y (a cor da curva é definido automaticamente)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordenadas X
   const double      &y[],          // coordenadas Y
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Versão para trabalhar com os pontos CPoint2D (a cor da curva é definido automaticamente)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordenadas dos pontos
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Versão para trabalhar com o ponteiro para a função CurveFunction (a cor da curva é definido automaticamente)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // ponteiro para a função
   const double      from,          // valor inicial do argumento
   const double      to,            // valor final do argumento
   const double      step,          // incremento do argumento
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

© 2000-2024, MetaQuotes Ltd.

Versão para trabalhar na coordenada Y (a cor da curva é definido pelo usuário)

```
CCurve*  CurveAdd(
   const double      &y[],          // coordenadas Y
   const uint        clr,           // cor da curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Observação

As coordenadas X para dada curva serão os índices de matriz Y.

Versão para trabalhar nas coordenadas Y e Y (a cor da curva é definido pelo usuário)

```
CCurve*  CurveAdd(
   const double      &x[],          // coordenadas X
   const double      &y[],          // coordenadas Y
   const uint        clr,           // cor da curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Versão para trabalhar com os pontos CPoint2D (a cor da curva é definido pelo usuário)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // coordenadas dos pontos
   const uint        clr,           // cor da curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

Versão para trabalhar com o ponteiro para a função CurveFunction (a cor da curva é definido pelo usuário)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // ponteiro para a função
   const double      from,          // valor inicial do argumento
   const double      to,            // valor final do argumento
   const double      step,          // incremento do argumento
   const uint        clr,           // cor da curva
   ENUM_CURVE_TYPE   type,          // tipo de curva
   const string      name=NULL      // nome da curva
   )
```

© 2000-2024, MetaQuotes Ltd.

Parâmetros

&x[]

[in] Coordenadas X.

&y[]

[in] Coordenadas Y.

&points[]

[in] Coordenadas dos pontos.

function

[in] Ponteiro para a função.

from

[in] Valor inicial de incremento do argumento.

to

[in] Valor final de incremento do argumento.

step

[in] Incremento do argumento.

type

[in] Tipo de curva.

name=NULL

[in] Nome da curva.

clr

[in] Cor da curva.

Valor de retorno

Ponteiro para a curva criada.

© 2000-2024, MetaQuotes Ltd.

CurvePlot

Desenha uma curva previamente criada de acordo com o índice.

```
bool  CurvePlot(
   const int  index      // índice
   )
```

Parâmetros

index

[in] Índice da curva

Valor de retorno

true no caso de sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

CurvePlotAll

Exibe todas as curvas adicionadas anteriormente ao gráfico.

```
bool  CurvePlotAll()
```

Valor de retorno

true no caso de sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

CurveGetByIndex

Obtém a curva no índice especificado.

```
CCurve*  CurveGetByIndex(
   const int  index      // índice da curva
   )
```

Parâmetros

index

[in] Índice da curva.

Valor de retorno

Ponteiro para a curva no índice definido.

© 2000-2024, MetaQuotes Ltd.

CurveGetByName

Obtém a curva de acordo com o nome especificado.

```
CCurve*  CurveGetByName(
   const string  name      // nome da curva
   )
```

Parâmetros

name

[In] Nome da curva.

Valor de retorno

Ponteiro para a primeira curva encontrada no índice definido.

© 2000-2024, MetaQuotes Ltd.

CurveRemoveByIndex

Exclui a curva segundo o índice especificado.

```
bool  CurveRemoveByIndex(
   const int  index      // índice da curva
   )
```

Parâmetros

index

[in] Índice da curva a ser excluída.

Valor de retorno

true — no caso de sucesso, caso contrário — false.

© 2000-2024, MetaQuotes Ltd.

CurveRemoveByName

Exclui a curva segundo o nome especificado.

```
bool  CurveRemoveByName(
   const string  name      // nome da curva
   )
```

Parâmetros

nome

[in] Nome da curva a ser excluída.

Valor de retorno

true — no caso de sucesso, caso contrário — false.

© 2000-2024, MetaQuotes Ltd.

CurvesTotal

Retorna o número de curvas para um determinado gráfico.

```
int  CurvesTotal()
```

Valor de retorno

Número de curvas.

Observação

São consideradas todas as curvas que pertencem a este gráfico, independentemente de seu estilo de desenho e visibilidade.

© 2000-2024, MetaQuotes Ltd.

MarksToAxisAdd

Adiciona as marcas de escala para o eixo definido no gráfico.

```
bool  MarksToAxisAdd(
   const double        &marks[],        // coordenadas das marcas
   const int           mark_size,       // tamanho das marcas
   ENUM_MARK_POSITION  position,        // localização das marcas
   const int           dimension=0      // dimensão
   )
```

Parâmetros

&marks[]

[in] Coordenadas das marcas

mark_size

[in] Tamanho das marcas

position

[in] Localização das marcas

dimension=0

[in] 0 — adição ao eixo X,

1 — adição ao eixo Y

Valor de retorno

true no caso de sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

MajorMarkSize (método Get)

Retorna o tamanho de marcas da escala no eixo de coordenadas.

```
int  MajorMarkSize()
```

MajorMarkSize (método Set)

Define o tamanho de marcas da escala no eixo de coordenadas.

```
void  MajorMarkSize(
   const int  size      // tamanho das marcas
   )
```

Parâmetros

size

[in] Tamanho das marcas em pixels.

© 2000-2024, MetaQuotes Ltd.

FontSet

Define os parâmetros da fonte atual.

```
bool  FontSet(
   const string  name,         // nome
   const int     size,         // tamanho
   const uint    flags=0,      // sinalizadores
   const uint    angle=0       // ângulo
   )
```

Parâmetros

name

[in] Nome.

size

[in] Tamanho.

flags=0

[in] Sinalizador.

angle=0

[in] Ângulo.

Valor de retorno

true — no caso de sucesso, caso contrário — false.

© 2000-2024, MetaQuotes Ltd.

FontGet

Obtém os parâmetros da fonte atual.

```
void  FontGet(
   string  &name,       // nome
   int     &size,       // tamanho
   uint    &flags,      // sinalizadores
   uint    &angle       // ângulo
   )
```

Parâmetros

&name

[out] Nome.

&size

[out] Tamanho.

&flags

[out] Sinalizadores.

&angle

[out] Ângulo.

© 2000-2024, MetaQuotes Ltd.

GridAxisLineColor (Método Get)

Retorna o valor de cor de eixos reais do gráfico.

```
uint  GridAxisLineColor()
```

Valor de retorno

Cor de eixos reais do gráfico.

GridAxisLineColor (Método Set)

Define o valor de cor de eixos reais do gráfico.

```
void  GridAxisLineColor(
   const uint  clr      // cor dos eixos do gráfico
   )
```

Parâmetros

clr

[in] Cor de eixos reais do gráfico.

© 2000-2024, MetaQuotes Ltd.

Attach

Versão para obter - a partir do objeto OBJ_BITMAP_LABEL - o objeto gráfico e sua vinculação à instância da classe CGraphic:

```
bool  Attach(
   const long    chart_id,      // identificador de gráfico
   const string  objname        // nome de objeto gráfico
   )
```

Versão para criar o objeto gráfico para o objeto OBJ_BITMAP_LABEL e sua vinculação à instância da classe CGraphic:

```
bool  Attach(
   const long    chart_id,      // identificador de gráfico
   const string  objname,       // nome de objeto gráfico
   const int     width,         // largura de imagem
   const int     height         // altura de imagem
   )
```

Parâmetros

chart_id

[in] Identificador do gráfico.

objname

[in] Título (nome) do objeto gráfico.

width

[in] Largura de imagem no recurso.

height

[in] Altura de imagem no recurso.

Valor de retorno

true — ser for bem-sucedido, false — se não for possível vincular o objeto.

© 2000-2024, MetaQuotes Ltd.

CalculateMaxMinValues

Executa cálculo (recálculo) dos valores mínimo e máximo do gráfico em ambos os eixos.

```
void  CalculateMaxMinValues()
```

© 2000-2024, MetaQuotes Ltd.

Height

Retorna a altura do gráfico em pixels.

```
int  Height()
```

Valor de retorno

Altura do gráfico em pixels.

© 2000-2024, MetaQuotes Ltd.

IndentDown (Método Get)

Retorna o recuo a partir da borda inferior do gráfico.

```
int  IndentDown()
```

Valor de retorno

Tamanho do recuo, em pixels.

IndentDown (Método Set)

Define o recuo a partir da borda inferior do gráfico.

```
void  IndentDown(
   const int  down      // tamanho do recuo
   )
```

Parâmetros

down

[in] Tamanho do recuo em pixels.

© 2000-2024, MetaQuotes Ltd.

IndentLeft (Método Get)

Retorna o recuo do gráfico a partir da borda esquerda.

```
int  IndentLeft()
```

Valor de retorno

Tamanho do recuo, em pixels.

IndentLeft (Método Set)

Define o recuo do gráfico a partir da borda esquerda.

```
void  IndentLeft(
   const int  left      // tamanho do recuo
   )
```

Parâmetros

left

[in] Tamanho do recuo em pixels.

© 2000-2024, MetaQuotes Ltd.

IndentRight (Método Get)

Retorna o recuo do gráfico a partir da borda direita.

```
int  IndentRight()
```

Valor de retorno

Tamanho do recuo, em pixels.

IndentRight (Método Set)

Define o recuo do gráfico a partir da borda direita.

```
void  IndentRight(
   const int  right      // tamanho do recuo
   )
```

Parâmetros

right

[in] Tamanho do recuo em pixels.

© 2000-2024, MetaQuotes Ltd.

IndentUp (Método Get)

Retorna o recuo do gráfico a partir da borda superior.

```
int  IndentUp()
```

Valor de retorno

Tamanho do recuo, em pixels.

IndentUp (Método Set)

Define o recuo do gráfico a partir da borda superior.

```
void  IndentUp(
   const int  up      // tamanho do recuo
   )
```

Parâmetros

up

[in] Valor do recuo em pixels.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redesenha o gráfico.

```
bool  Redraw(
   const bool  rescale=false      // valor do sinalizador
   )
```

Parâmetros

rescale=false

[in] Sinalizador que indica se é necessário redimensionar o gráfico.

Valor de retorno

true no caso de sucesso, caso contrário false.

© 2000-2024, MetaQuotes Ltd.

ResetParameters

Restaura os parâmetros necessários para redesenhar o gráfico.

```
void ResetParameters()
```

© 2000-2024, MetaQuotes Ltd.

SetDefaultParameters

Define os parâmetros do gráfico para os valores padrão.

```
void  SetDefaultParameters()
```

© 2000-2024, MetaQuotes Ltd.

ScaleX

Dimensiona o valor no eixo X.

```
virtual int  ScaleX(
   double  x      // valor do eixo X
   )
```

Parâmetros

x

[in] Valore real do eixo X.

Valor de retorno

Valor em pixels.

Observação

Dimensiona o valor real em os pixels para exibição no gráfico.

© 2000-2024, MetaQuotes Ltd.

ScaleY

Dimensiona o valor no eixo Y.

```
virtual int  ScaleY(
   double  y      // valor do eixo y
   )
```

Parâmetros

y

[in] Valore real do eixo Y.

Valor de retorno

Valor em pixels.

Observação

Dimensiona o valor real em os pixels para exibição no gráfico.

© 2000-2024, MetaQuotes Ltd.

Width

Retorna a largura do gráfico em pixels.

```
int  Width()
```

Valor de retorno

Largura do gráfico em pixels.

© 2000-2024, MetaQuotes Ltd.

Indicadores Técnicos e TimeSeries

Esta seção contém os detalhes técnicos das clasee para indicador técnico e timeSeries, e também, a descrição dos componentes correspondentes na biblioteca padrão MQL5.

A utilização das classes de indicador técnico e timeSeries irá poupar tempo no desenvolvimento de aplicativos (scripts, Expert Advisors).

A Biblioteca Padrão MQL5 é alocada no diretório de trabalho do terminal, na pasta Include\Indicators.

Classe/grupo	Descrição
Classes Base	Grupo de base e classes auxiliares
Classes de timesseries	Grupo de classes timeSeries
Indicadores de Tendência	Grupo de classes de indicador de tendência
Osciladores	Grupo de classes de indicador oscilador
Indicadores de Volume	Grupo de classes de indicador de volume
Indicadores Bill Williams	Classes do indicador Bill Williams
indicadores personalizados	Classe do indicador personalizado

© 2000-2024, MetaQuotes Ltd.

Base e Indicador Técnico Auxiliar e Classes de TimeSeries

Esta seção contém os detalhes técnicos de base, indicador técnico auxiliar, classes de timeSeries e a descrição dos componentes correspondentes na biblioteca padrão MQL5.

Classe/grupo	Descrição
CSpreadBuffer	Histórico da classe de buffer do spread
CTimeBuffer	Histórico da classe de buffer dos tempos de abertura das barras
CTickVolumeBuffer	Histórico da classe de buffer dos volumes de ticks
CRealVolumeBuffer	Histórico da classe de buffer dos volumes reais
CDoubleBuffer	Classe base do tipo double de buffer de dados
COpenBuffer	Classe de buffer da abertura de preço na barra
CHighBuffer	Classe de buffer da máxima de preço na barra
CLowBuffer	Classe de buffer da mínima de preço na barra
CCloseBuffer	Classe de buffer do fechamento de preço na barra
CIndicatorBuffer	Classe de buffer de indicador técnico
CSeries	Classe de base para o acesso aos dados de timeseries
CPriceSeries	Classe de base para o acesso a dados dos preços
CIndicator	Classe de base do indicador técnico
CIndicators	Indicador técnico e coleção de séries

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

CSpreadBuffer

CSpreadBuffer é uma classe para o acesso simplificado aos spreads das barras no histórico.

Descrição

A classe CSpreadBuffer fornece um acesso aos spreads das barras no histórico.

Declaração

```
class CSpreadBuffer: public CArrayInt
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayInt
      CSpreadBuffer

Métodos de classe

Atributos	
Size	Define o tamanho do buffer
Configuração	
SetSymbolPeriod	Define símbolo e período
Métodos de Acesso aos Dados	
At	Obtém o elemento de buffer pelo índice
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Define o tamanho do buffer.

```
void  Size(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Define símbolo e período.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbol
   const ENUM_TIMEFRAMES  period      // period
   )
```

Parâmetros

symbol

[in] Novo símbolo.

period

[in] Novo período (enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento de buffer pelo índice.

```
int  At(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CTimeBuffer

CTimeBuffer é uma classe de acesso simplificado aos tempos de abertura das barras no histórico.

Descrição

A classe CTimeBuffer providencia um acesso as aberturas dos tempos das barras no histórico.

Declaração

```
class CTimeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayLong
      CTimeBuffer

Métodos de classe

Atributos	
Size	Define o tamanho do buffer
Configuração	
SetSymbolPeriod	Define símbolo e período
Métodos de Acesso aos Dados	
At	Obtém o elemento de buffer pelo índice
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Define o tamanho do buffer.

```
void  Size(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Define símbolo e período.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbol
   const ENUM_TIMEFRAMES  period      // period
   )
```

Parâmetros

symbol

[in] Novo símbolo.

period

[in] Novo período (enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento de buffer pelo índice.

```
long  At(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CTickVolumeBuffer

CTickVolumeBuffer é uma classe de acesso simplificado aos volumes de ticks das barras no histórico.

Descrição

A classe CTickVolumeBuffer fornece um acesso aos volumes de ticks das barras no histórico.

Declaração

```
class CTickVolumeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

Métodos de classe

Atributos	
Size	Define o tamanho do buffer
Configuração	
SetSymbolPeriod	Define símbolo e período
Métodos de Acesso aos Dados	
At	Obtém o elemento de buffer pelo índice
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Define o tamanho do buffer.

```
void  Size(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Define símbolo e período.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbol
   const ENUM_TIMEFRAMES  period      // period
   )
```

Parâmetros

symbol

[in] Novo símbolo.

period

[in] Novo período (enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento de buffer pelo índice.

```
long  At(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CRealVolumeBuffer

CRealVolumeBuffer é uma classe de acesso simplificado aos volumes reais das barras no histórico.

Descrição

A classe CTickVolumeBuffer providencia um acesso a volumes reais das barras no histórico.

Declaração

```
class CRealVolumeBuffer: public CArrayLong
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

Métodos de classe

Atributos	
Size	Define o tamanho do buffer
Configuração	
SetSymbolPeriod	Define símbolo e período
Métodos de Acesso aos Dados	
At	Obtém o elemento de buffer pelo índice
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Define o tamanho do buffer.

```
void  Size(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Define símbolo e período.

```
void  SetSymbolPeriod(
   const string           symbol,     // symbol
   const ENUM_TIMEFRAMES  period      // period
   )
```

Parâmetros

symbol

[in] Novo símbolo.

period

[in] Novo período (enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento de buffer pelo índice.

```
long  At(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CDoubleBuffer

CDoubleBuffer é uma classe base de acesso simplificado para buffers de dados do tipo double.

Descrição

A classe CDoubleBuffer proporciona um acesso para buffers de dados do tipo double

Declaração

```
class CDoubleBuffer: public CArrayDouble
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

Descendentes diretos

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

Métodos de classe

Atributos	
Size	Define o tamanho do buffer
Configuração	
SetSymbolPeriod	Define símbolo e período
Métodos de Acesso aos Dados	
At	Obtém o elemento de buffer
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2024, MetaQuotes Ltd.

Size

Define o tamanho do buffer.

```
void  Size(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

SetSymbolPeriod

Define símbolo e período.

```
void  SetSymbolPeriod(
   const string           symbol,      // symbol
   const ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Novo símbolo.

period

[in] Novo período (enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

At

Obtém o elemento de buffer pelo índice.

```
double  At(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

COpenBuffer

COpenBuffer é uma classe para o acessar simplificadamente a abertura de preços das barras no histórico.

Descrição

A classe COpenBuffer tem por objetivo acessar a abertura de preços das barras no histórico.

Declaração

```
class COpenBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        COpenBuffer

Métodos de classe

Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos herdados da classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CHighBuffer

CHighBuffer é uma classe de acesso simplificado as máximas de preços das barras no histórico.

Descrição

A classe CHighBuffer fornece um acesso as máximas de preços das barras no histórico.

Declaração

```
class CHighBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

Métodos de classe

Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos herdados da classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CLowBuffer

CLowBuffer é uma classe para acessar simplificadamente a mínima de preços das barras no histórico.

Descrição

A classe CLowBuffer tem por objetivo acessar a mínima de preços das barras no histórico.

Declaração

```
class CLowBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

Métodos de classe

Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos herdados da classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CCloseBuffer

CCloseBuffer é uma classe de acesso simplificado para fechamento de preços das barras no histórico.

Descrição

A classe CCloseBuffer fornece o acesso para fechamento de preços das barras no histórico.

Declaração

```
class CCloseBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

Métodos de classe

Métodos de Atualização dos Dados	
virtual Refresh	Atualiza o buffer
virtual RefreshCurrent	Atualiza o valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos herdados da classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o elemento atual (zeroth) do buffer.

```
virtual bool  RefreshCurrent()
```

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CIndicatorBuffer

CIndicatorBuffer é uma classe para o acesso simplificado aos dados de buffer do indicador.

Descrição

A classe CIndicatorBuffer fornece o acesso simplificado aos dados de buffer do indicador técnico.

Declaração

```
class CIndicatorBuffer: public CDoubleBuffer
```

Título

```
#include <Indicators\Indicator.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CIndicatorBuffer

Métodos de classe

Atributos	
Offset	Obtém/Define deslocamento do buffer
Name	Obtém/Define nome do buffer
Métodos de Acesso aos Dados	
At	Obtém elemento de buffer
Métodos de Atualização dos Dados	
Refresh	Atualiza o buffer
RefreshCurrent	Atualizações apenas de valor atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Métodos herdados da classe CDoubleBuffer

Size, At, SetSymbolPeriod

© 2000-2024, MetaQuotes Ltd.

Offset

Obtém deslocamento do buffer.

```
int  Offset() const
```

Valor de retorno

Deslocamento do buffer.

Offset

Define deslocamento do buffer.

```
void  Offset(
   const int  offset      // offset
   )
```

Parâmetros

offset

[in] Deslocamento do novo buffer

© 2000-2024, MetaQuotes Ltd.

Name

Obtém o nome do buffer.

```
string  Name() const
```

Valor de retorno

Nome do buffer.

Name

Define o nome do buffer.

```
void  Name(
   const string  name      // name
   )
```

Parâmetros

name

[in] Novo nome do buffer.

© 2000-2024, MetaQuotes Ltd.

At

Obtém elemento de buffer pelo índice.

```
double  At(
   int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

Elemento de buffer com o índice especificado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza o buffer por completo.

```
bool  Refresh(
   const int  handle,      // handle
   const int  num          // buffer number
   )
```

Parâmetros

handle

[in] Manipulador do indicador.

num

[in] Índice de buffer do indicador.

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Atualiza o atual elemento de buffer (zeroth).

```
bool  RefreshCurrent(
   const int  handle,      // handle of the indicator
   const int  num          // buffer number
   )
```

Parâmetros

handle

[in] Manipulador do indicador.

num

[in] Número de Buffer.

Valor de retorno

verdadeiro - se bem sucedido; falso - se o buffer não foi atualizado.

© 2000-2024, MetaQuotes Ltd.

CSeries

CSeries é uma classe base para o acesso a dados das timeseries da Biblioteca Padrão.

Descrição

A classe CSeries fornece o acesso simplificado as funções TimeSeries MQL5 para todos os seus descendentes.

Declaração

```
class CSeries: public CArrayObj
```

Título

```
#include <Indicators\Series.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries

Descendentes diretos

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

Métodos de classe

Atributos	
Name	Obtém o nome das timeseries ou indicador
BuffersTotal	Obtém o número dos buffers das timeseries ou indicador
Timeframe	Obtém o flag do timeframe das timeseries ou indicador
Symbol	Obtém o símbolo da timeseries ou indicador
Period	Obtém o período das timeseries ou indicador
RefreshCurrent	Obtém/Define o flag de atualização dos dados atuais
Métodos de Acesso aos Dados	
virtual BufferResize	Define o tamanho do buffer das timeseries ou indicador
Métodos de Atualização dos Dados	
virtual Refresh	Atualizar os dados das timeseries ou indicador
PeriodDescription	Obtém o período como uma string

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Name

Obtém o nome das timeseries ou indicador

```
string  Name() const
```

Valor de retorno

O nome das timeseries ou indicador.

© 2000-2024, MetaQuotes Ltd.

BuffersTotal

Obtém o número dos buffers das timeseries ou indicador.

```
int  BuffersTotal() const
```

Valor de retorno

O número dos buffers das timeseries ou indicador.

Observação

As timeseries tem um único buffer.

© 2000-2024, MetaQuotes Ltd.

Timeframe

Obtém o flag do timeframe das timeseries ou indicador.

```
int  Timeframe() const
```

Valor de retorno

O flag do timeframe das timeseries ou indicador.

Observação

É o flag de visibilidade de alguns timeframes.

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém o símbolo (ativo) das timeseries ou indicador.

```
string  Symbol() const
```

Valor de retorno

O símbolo das timeseries ou indicador.

© 2000-2024, MetaQuotes Ltd.

Period

Obtém o período das timeseries ou indicador.

```
ENUM_TIMEFRAMES  Period() const
```

Valor de retorno

O período das timeseries ou indicador (valor do enumerador ENUM_TIMEFRAMES).

© 2000-2024, MetaQuotes Ltd.

RefreshCurrent

Define um flag para atualizar os valores atuais das timeseries ou indicador.

```
string  RefreshCurrent(
   const bool  flag      // new flag
   )
```

Parâmetros

flag

[in] Novo flag

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

BufferSize

Retorna a quantidade de dados disponíveis no buffer das timeseries ou indicador.

```
int  BufferSize() const
```

Valor de retorno

Quantidade de dados disponíveis no buffer das timeseries ou indicador.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define o tamanho do buffer das timeseries ou indicador.

```
virtual bool  BufferResize(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho dos buffers.

Valor de retorno

verdadeiro - se bem sucedido, falso - do contrário.

Observação

Todos os buffers das timeseries ou indicadores têm o mesmo tamanho.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries ou indicador.

```
virtual void  Refresh(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Timeframes para atualizar (flag).

© 2000-2024, MetaQuotes Ltd.

PeriodDescription

Obtém a representação da string do enumerador especificado ENUM_TIMEFRAMES.

```
string  PeriodDescription(
   const int  val=0      // value
   )
```

Parâmetros

val=0

[in] Valor para converter.

Valor de retorno

A representação da string do enumerador ENUM_TIMEFRAMES.

Observação

Se o valor não for especificado ou for igual a zero, retorna o prazo das timeseries ou indicador.

© 2000-2024, MetaQuotes Ltd.

CPriceSeries

CPriceSeries é uma classe base para acesso aos dados do preço.

Descrição

A classe CSeries fornece o acesso as funções MQL5 para trabalhar com dados de preços para todos os seus descendentes.

Declaração

```
class CPriceSeries: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries

Descendentes diretos

CiClose, CiHigh, CiLow, CiOpen

Métodos de classe

Métodos de Criação	
virtual BufferResize	Define o tamanho do buffer
Métodos de Acesso aos Dados	
virtual GetData	Obtém o elemento de buffer especificado pelo índice
Métodos de Atualização dos Dados	
virtual Refresh	Atualiza dados das timeseries
Métodos de Pesquisa de Dados	
virtual MinIndex	Obtém o índice do elemento mínimo no intervalo especificado
virtual MinValue	Obtém o valor e índice do elemento mínimo no intervalo especificado
virtual MaxIndex	Obtém o índice do elemento máximo no intervalo especificado
virtual MaxValue	Obtém o valor e índice do elemento máximo no intervalo especificado

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define novo tamanho do buffer.

```
virtual void  BufferResize(
   const int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento de buffer especificado pelo índice.

```
double  GetData(
   const int  index      // index
   ) const
```

Parâmetros

index

[in] Índice do elemento de buffer.

Valor de retorno

O elemento de buffer com o índice especificado ou EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // timeframe flags
   )
```

Parâmetros

flags=OBJ_ALL_PERIODS

[in] Timeframes para atualizar (flag).

© 2000-2024, MetaQuotes Ltd.

MinIndex

Obtém o índice do elemento mínimo no intervalo especificado.

```
virtual int  MinIndex(
   const int  start,      // starting index
   const int  count       // number of elements to scan
   ) const
```

Parâmetros

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

Valor de retorno

O índice do elemento mínimo no intervalo especificado, ou -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

MinValue

Obtém o valor e índice do elemento mínimo no intervalo especificado.

```
virtual double  MinValue(
   const int   start,     // starting index
   const int   count,     // number of elements to scan
   int&        index      // reference to the variable for index
   ) const
```

Parâmetros

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

index

[out] Referente à variável do tipo inteiro.

Valor de retorno

O valor do elemento mínimo do buffer no intervalo especificado, ou EMPTY_VALUE em caso de erro.

Observação

O índice do elemento mínimo é armazenado na variável index.

© 2000-2024, MetaQuotes Ltd.

MaxIndex

Obtém o índice do elemento máximo em intervalo especificado.

```
virtual int  MaxIndex(
   const int  start,      // starting index
   const int  count       // number of elements to scan
   ) const
```

Parâmetros

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

Valor de retorno

O índice do elemento máximo em intervalo especificado, ou -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

MaxValue

Obtém o valor e o índice do elemento máximo no intervalo especificado.

```
virtual double  MaxValue(
   const int   start,     // starting index
   const int   count,     // number of elements to scan
   int&        index      // reference to the variable for index
   ) const
```

Parâmetros

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

index

[out] Referente à variável do tipo inteiro.

Valor de retorno

O valor do elemento máximo do buffer especificado no intervalo, ou EMPTY_VALUE em caso de erro.

Observação

O índice do elemento máximo é armazenado na variável index.

© 2000-2024, MetaQuotes Ltd.

CIndicator

CIndicator é uma classe base para indicadores técnicos da biblioteca padrão MQL .

Descrição

A classe CIndicator possibilita o acesso simplificado a todos os seus descendentes das funções dos indicadores técnicos API MQL5.

Declaração

```
class CIndicator: public CSeries
```

Título

```
#include <Indicators\Indicator.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator

Descendentes diretos

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

Métodos de classe

Atributos	
Handle	Obtém o manipulador do indicador.
Status	Obtém o estado do indicador.
FullRelease	Define um flag para liberar o manipulador do indicador.
Criação	
Create	Cria os indicadores
BufferResize	Define novo tamanho de buffer
Métodos de Acesso aos Dados	
GetData	Copiando os dados do buffer de indicador
Métodos de Atualização dos Dados	

© 2000-2024, MetaQuotes Ltd.

Atributos	
Refresh	Atualiza dados do indicador
Encontrando Valores Min/Máx	
Minimum	Obtém o índice do elemento mínimo de buffer especificado em intervalo determinado.
MinValue	Obtém o valor e índice do elemento mínimo de buffer especificado num intervalo determinado.
Maximum	Obtém o índice do elemento máximo de buffer especificado em intervalo determinado.
MaxValue	Obtém o valor e índice do elemento máximo de buffer especificado em intervalo determinado
Conversão de Enumeradores	
MethodDescription	Obtém o valor do enumerador ENUM_MA_METHOD como string
PriceDescription	Obtém o valor do enumerador ENUM_APPLIED_PRICE como string
VolumeDescription	Obtém o valor do enumerador ENUM_APPLIED_VOLUME como string
Trabalhando com gráficos	
AddToChart	Adiciona o indicador no gráfico
DeleteFromChart	Exclui o indicador do gráfico

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

As classes derivadas:

- CiAC
- CiAD
- CiADX

© 2000-2024, MetaQuotes Ltd.

- CiADXWilder
- CiAlligator
- CiAMA
- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
- CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA
- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes
- CiWPR

© 2000-2024, MetaQuotes Ltd.

Handle

Obtém o manipulador do indicador.

```
int  Handle() const
```

Valor de retorno

Manipulador do indicador.

© 2000-2024, MetaQuotes Ltd.

Status

Obtém o estado do indicador.

```
string  Status() const
```

Valor de retorno

O estado de criação do indicador.

© 2000-2024, MetaQuotes Ltd.

FullRelease

Define um flag para liberar o manipulador do indicador.

```
void  FullRelease(
   const bool  flag=true      // flag
   )
```

Parâmetros

flag

[in] O novo valor do flag para liberar o manipulador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com os parâmetros determinados.

```
bool  Create(
   const string            symbol,          // symbol
   const ENUM_TIMEFRAMES   period,          // period
   const ENUM_INDICATOR    type,            // type
   const int               num_params,      // number of parameters
   const MqlParam&         params[]         // reference to the parameters array
   )
```

Parâmetros

symbol

[in] Nome do ativo.

period

[in] Período (enumerador ENUM_TIMEFRAMES).

type

[in] tipo de indicador (enumerador ENUM_INDICATOR).

num_params

[in] Número de parâmetros do indicador.

params

[in] Referente aos parâmetros do array do indicador.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define o tamanho do buffer do indicador.

```
virtual bool  BufferResize(
   const int  size      // size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

Valor de retorno

verdadeiro - se bem sucedido, falso - do contrário.

Observação

Todos os buffers do indicador tem o mesmo tamanho.

© 2000-2024, MetaQuotes Ltd.

BarsCalculated

Retorna o número de dados calculados para o indicador.

```
int  BarsCalculated() const;
```

Valor do Retorno

Retorna a quantidade de dados calculados no buffer de indicador ou -1 em caso de erro (dados ainda não calculados).

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento específico do buffer determinado do indicador. Refresh() deve ser chamado para trabalhar com dados recentes antes de usar o método.

```
double  GetData(
   const int  buffer_num,      // buffer number
   const int  index            // element index
   ) const
```

Parâmetros

buffer_num

[in] Número de Buffer.

index

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento, ou EMPTY_VALUE em caso de erro.

GetData

Obtém os dados de buffer do indicador pela posição inicial e o número de dados necessários.

```
int  GetData(
   const int      start_pos,      // position
   const int      count,          // number of elements needed
   const int      buffer_num,     // buffer number
   double&        buffer[]        // target array for data
   ) const
```

Parâmetros

start_pos

[in] posição inicial do buffer do indicador.

count

[in] Número de elementos necessários.

buffer_num

[in] Número do buffer do indicador.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

Se com sucesso, retorna o número de elementos recebidos do buffer do indicador especificado, caso contrário, -1.

GetData

© 2000-2024, MetaQuotes Ltd.

Obtém os dados de buffer do indicador pela hora de início e o número de dados necessários.

```
int  GetData(
   const datetime  start_time,      // starting time
   const int       count,           // number of elements needed
   const int       buffer_num,      // buffer number
   double&         buffer[]         // target array for data
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer_num

[in] Número do buffer do indicador.

buffer

[in] Referente ao array de destino.

Valor de retorno

Se com sucesso, retorna o número de elementos recebidos do buffer do indicador especificado, caso contrário, -1.

GetData

Obtém os dados do buffer do indicador no início, no tempo final e o número de dados necessários.

```
int  GetData(
   const datetime  start_time,      // start time
   const datetime  stop_time,       // stop time
   const int       buffer_num,      // number of buffer
   double&         buffer[]         // target array for data
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer_num

[in] Número do buffer do indicador.

buffer

[in] Referente ao array de destino.

© 2000-2024, MetaQuotes Ltd.

Valor de retorno

Se com sucesso, retorna o número de elementos recebidos do buffer do indicador especificado, caso contrário, -1.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza dados do indicador. É recomendado chamar o método antes de usar GetData().

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // flags
   )
```

Parâmetros

flags=OBJ_ALL_PERIODS

[in] Timeframe de atualização dos flags.

© 2000-2024, MetaQuotes Ltd.

Minimum

Retorna o índice do elemento mínimo do buffer especificado em intervalo determinado.

```
int  Minimum(
   const int  buffer_num,      // buffer number
   const int  start,           // starting index
   const int  count            // number of elements to proceed
   ) const
```

Parâmetros

buffer_num

[in] Número de Buffer para procurar o valor.

start

[in] Iniciando o índice de pesquisa.

count

[in] Número de elementos a pesquisar.

Valor de retorno

Índice do elemento mínimo do buffer especificado em intervalo determinado.

© 2000-2024, MetaQuotes Ltd.

MinValue

Retorna o valor e índice de elemento mínimo do buffer especificado em intervalo determinado.

```
double  MinValue(
   const int   buffer_num,      // buffer number
   const int   start,           // starting index
   const int   count,           // number of elements to proceed
   int&        index            // reference
   ) const
```

Parâmetros

buffer_num

[in] Número de Buffer para procurar o valor.

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

index

[out] Referente a variável do tipo int para o índice de elemento mínimo.

Valor de retorno

O valor do elemento mínimo do buffer especificado em intervalo determinado.

Observação

O índice do elemento mínimo do buffer é armazenado dentro do índice variável, transmitido pela referência.

© 2000-2024, MetaQuotes Ltd.

Maximum

Retorna o índice do elemento máximo do buffer especificado em intervalo determinado.

```
int  Maximum(
   const int  buffer_num,      // buffer number
   const int  start,           // starting index
   const int  count            // number of elements to proceed
   ) const
```

Parâmetros

buffer_num

[in] Número de Buffer para procurar o valor.

start

[in] Iniciando o índice de pesquisa.

count

[in] Número de elementos a pesquisar.

Valor de retorno

Índice do elemento máxima do buffer especificado em intervalo determinado.

© 2000-2024, MetaQuotes Ltd.

MaxValue

Retorna o valor e o índice de elemento máximo do buffer especificado em intervalo determinado.

```
double  MaxValue(
   const int   buffer_num,      // buffer number
   const int   start,           // starting index
   const int   count,           // number of elements to proceed
   int&        index            // reference
   ) const
```

Parâmetros

buffer_num

[in] Número de Buffer para procurar o valor.

start

[in] Iniciando o índice.

count

[in] Número de elementos para proceder.

index

[out] Referente a variável do tipo int para o índice do elemento máximo.

Valor de retorno

O valor do elemento máximo do buffer especificado em intervalo determinado.

Observação

O índice do elemento de buffer máximo é armazenado dentro do índice variável, transmitido por referência.

© 2000-2024, MetaQuotes Ltd.

MethodDescription

A função retorna o valor do enumerador ENUM_MA_METHOD como uma string.

```
string  MethodDescription(
   const int  val      // value
   ) const
```

Parâmetros

val

[in] Valor do enumerador ENUM_MA_METHOD.

Valor de retorno

O valor do enumerador ENUM_MA_METHOD como uma string.

© 2000-2024, MetaQuotes Ltd.

PriceDescription

O método retorna o valor do enumerador ENUM_APPLIED_PRICE como uma string.

```
string  PriceDescription(
   const int  val      // value
   ) const
```

Parâmetros

val

[in] Valor do enumerador ENUM_APPLIED_PRICE.

Valor de retorno

O valor do enumerador ENUM_APPLIED_PRICE como uma string.

© 2000-2024, MetaQuotes Ltd.

VolumeDescription

O método retorna o valor do enumerador ENUM_APPLIED_VOLUME como uma string.

```
string  VolumeDescription(
   const int  val      // value
   ) const
```

Parâmetros

val

[in] Valor do enumerador ENUM_APPLIED_VOLUME.

Valor de retorno

O valor do enumerador ENUM_APPLIED_VOLUME como uma string.

© 2000-2024, MetaQuotes Ltd.

AddToChart

Adiciona o indicador no gráfico.

```
bool  AddToChart(
   const long  chart,      // chart ID
   const int  subwin       // chart subwindow
   )
```

Parâmetros

chart

[in] ID do Gráfico

subwin

[in] Sub-janela do gráfico.

Valor de retorno

verdadeiro - se bem sucedido; falso - se com erro.

© 2000-2024, MetaQuotes Ltd.

DeleteFromChart

Exclui o indicador do gráfico.

```
bool  DeleteFromChart(
   const long  chart,      // chart ID
   const int  subwin       // chart subwindow
   )
```

Parâmetros

chart

[in] ID do Gráfico

subwin

[in] Sub-janela do gráfico.

Valor de retorno

verdadeiro - se bem sucedido; falso - se com erro.

© 2000-2024, MetaQuotes Ltd.

CIndicators

O CIndicators é uma classe para recolher instâncias de timeseries e classes de indicadores técnicos.

Descrição

A classe CIndicators possibilita a criação das instâncias de classe dos indicadores técnicos, o seu armazenamento e gestão (sincronização de dados, manipulação e gerenciamento de memória).

Declaração

```
class CIndicators: public CArrayObj
```

Título

```
#include <Indicators\Indicators.mqh>
```

Métodos de classe

Métodos de Criação	
Create	Cria indicador técnico
Métodos de Atualização dos Dados	
Refresh	Atualiza dados para todos os indicadores técnicos na coleção

© 2000-2024, MetaQuotes Ltd.

Create

Ele cria o indicador com os parâmetros especificados.

```
CIndicator*  Create(
   const string            symbol,      // Symbol name
   const ENUM_TIMEFRAMES   period,      // Period
   const ENUM_INDICATOR    type,        // Indicator's type
   const int               count,       // Number of parameters
   const MqlParam&         params       // Parameters array reference
   )
```

Parâmetros

symbol

[in] Nome do ativo.

period

[in] Período (enumerador ENUM_TIMEFRAMES).

type

[in] Tipo de indicador (ENUM_INDICATOR).

count

[in] Número de parâmetros para o indicador.

params

[in] Referente aos parâmetros do array do indicador.

Valor de retorno

Se for bem sucedido, retorna a referência do indicador criado, e NULL se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza dados para todos os indicadores técnicos na coleção.

```
int  Refresh()
```

Valor de retorno

Retorna a atualização dos flags de timeframe (formado como um objeto flags de visibilidade).

© 2000-2024, MetaQuotes Ltd.

Classes de timesseries

Este grupo de capítulos contém detalhes técnicos das classes TimeSeries da biblioteca padrão MQL5 e descrições de todos os seus componentes fundamentais.

Classe	Descrição
CiSpread	Fornece um acesso aos dados históricos do spread
CiTime	Fornece um acesso aos tempos de abertura das barras no histórico
CiTickVolume	Fornece um acesso aos volumes de ticks das barras no histórico
CiRealVolume	Fornece um acesso aos volumes reais das barras no histórico
CiOpen	Fornece um acesso para abertura dos preços das barras no histórico
CiHigh	Fornece um acesso para máxima dos preços das barras no histórico
CiLow	Fornece um acesso para mínima dos preços das barras no histórico
CiClose	Fornece um acesso ao fechamento dos preços das barras no histórico

© 2000-2024, MetaQuotes Ltd.

CiSpread

CiSpread é uma classe projetada para acesso aos spreads das barras no histórico.

Descrição

A classe CiSpread fornece um acesso aos spreads de dados no histórico.

Declaração

```
class CiSpread: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CiSpread

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
BufferResize	Define o tamanho do buffer
Métodos de Acesso aos Dados	
GetData	Obtém os dados
Métodos de Atualização dos Dados	
Refresh	Atualiza os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para acesso ao histórico dos spreads.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define novo tamanho das série.

```
virtual void  BufferResize(
   int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pelo índice.

```
int  GetData(
   int  index       // index
   ) const
```

Parâmetros

index

[in] O índice do elemento necessário.

Valor de retorno

Elemento de buffer das timeseries, ou 0.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   int&  buffer          // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   int&      buffer           // target array
   ) const
```

Parâmetros

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   int&      buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags de timeframe

© 2000-2024, MetaQuotes Ltd.

CiTime

CiTime é uma classe projetada de acesso aos tempos de abertura das barras no histórico.

Descrição

A classe CiTime fornece um acesso aos tempos de abertura das barras no histórico.

Declaração

```
class CiTime: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CiTime

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
BufferResize	Define o tamanho do buffer
Métodos de Acesso aos Dados	
GetData	Obtém os dados
Métodos de Atualização dos Dados	
Refresh	Atualiza os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados de acesso aos tempos de abertura das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define novo tamanho das série.

```
virtual void  BufferResize(
   int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pelo índice.

```
datetime  GetData(
   int  index      // index
   ) const
```

Parâmetros

index

[in] O índice do elemento necessário.

Valor de retorno

Elemento de buffer das timeseries, ou 0.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   long& buffer          // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags de timeframe

© 2000-2024, MetaQuotes Ltd.

CiTickVolume

CiTickVolume é uma classe projetada para acesso aos volumes de ticks das barras no histórico.

Descrição

A classe CiTickVolume fornece um acesso aos volumes de ticks das barras no histórico.

Declaração

```
class CiTickVolume: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CiTickVolume

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
BufferResize	Define o tamanho do buffer
Métodos de Acesso aos Dados	
GetData	Obtém os dados
Métodos de Atualização dos Dados	
Refresh	Atualiza os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para acesso aos volumes de ticks das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define novo tamanho das série.

```
virtual void  BufferResize(
   int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pelo índice.

```
long  GetData(
   int  index       // index
   ) const
```

Parâmetros

index

[in] O índice do elemento necessário.

Valor de retorno

Elemento de buffer das timeseries, ou 0.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   long& buffer          // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags de timeframe

© 2000-2024, MetaQuotes Ltd.

CiRealVolume

CiRealVolume é uma classe projetada para acesso aos volumes reais das barras no histórico.

Descrição

A classe CiRealVolume fornece um acesso aos volumes reais das barras no histórico.

Declaração

```
class CiRealVolume: public CSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CiRealVolume

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
BufferResize	Define o tamanho do buffer
Métodos de Acesso aos Dados	
GetData	Obtém os dados
Métodos de Atualização dos Dados	
Refresh	Atualiza os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para acesso aos volumes reais das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

BufferResize

Define novo tamanho das série.

```
virtual void  BufferResize(
   int  size      // new size
   )
```

Parâmetros

size

[in] Novo tamanho do buffer

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pelo índice.

```
datetime  GetData(
   int  index      // index
   ) const
```

Parâmetros

index

[in] O índice do elemento necessário.

Valor de retorno

Elemento de buffer das timeseries, ou 0.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   long& buffer          // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

© 2000-2024, MetaQuotes Ltd.

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   long&     buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados das timeseries.

```
virtual void  Refresh(
   int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags de timeframe

© 2000-2024, MetaQuotes Ltd.

CiOpen

CiOpen é uma classe projetada para acessar a abertura dos preços das barras no histórico.

Descrição

A classe CiOpen fornece o acesso da abertura dos preços das barras no histórico.

Declaração

```
class CiOpen: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
Métodos de Acesso aos Dados	
GetData	Obtém os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados de acesso a abertura dos preços das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   double& buffer        // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   double&     buffer         // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   double&   buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

CiHigh

CiHigh é uma classe projetado para o acesso as máximas de preços das barras no histórico.

Descrição

A classe CiHigh fornece um acesso as máximas de preços das barras no histórico.

Declaração

```
class CiHigh: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiHigh

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
Métodos de Acesso aos Dados	
GetData	Obtém os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para acesso as máximas de preços das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   double& buffer        // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   double&     buffer         // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   double&   buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

CiLow

CiLow é uma classe projetado para o acesso as mínimas dos preços das barras no histórico.

Descrição

A classe CiLow fornece um acesso as mínimas dos preços das barras no histórico.

Declaração

```
class CiLow: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiLow

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
Métodos de Acesso aos Dados	
GetData	Obtém os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para o acesso as mínimas dos preços das barras no histórico.

```
bool  Create(
   string           symbol,      // symbol
   ENUM_TIMEFRAMES  period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   double& buffer        // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   double&     buffer         // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   double&   buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

CiClose

CiClose é uma classe projetada para acesso ao fechamento dos preços das barras no histórico.

Descrição

A classe CiClose fornece um acesso ao fechamento dos preços das barras no histórico.

Declaração

```
class CiClose: public CPriceSeries
```

Título

```
#include <Indicators\TimeSeries.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiClose

Métodos de classe

Métodos de Criação	
Create	Cria timeseries
Métodos de Acesso aos Dados	
GetData	Obtém os dados

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2024, MetaQuotes Ltd.

Create

Cria timeseries com os parâmetros especificados para acessar o fechamento de preços das barras no histórico.

```
bool  Create(
   string            symbol,      // symbol
   ENUM_TIMEFRAMES   period       // period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se as timeseries não foram criadas.

© 2000-2024, MetaQuotes Ltd.

GetData

Obtém o elemento das timeseries pela posição inicial e número de elementos.

```
int  GetData(
   int   start_pos,      // starting position
   int   count,          // number of elements to get
   double& buffer        // target array
   ) const
```

Parâmetros

start_pos

[in] Posição inicial das timeseries.

count

[in] Número de elementos necessários.

buffer

[in] Referência do array de destino aos dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início do tempo e pelo número de elementos.

```
int  GetData(
   datetime  start_time,      // starting time
   int       count,           // number of elements
   double&     buffer         // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

count

[in] Número de elementos necessários.

buffer

[in] Referente ao array de destino para os dados.

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

GetData

Obtém o elemento das timeseries pelo início e final dos tempos.

© 2000-2024, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,      // starting time
   datetime  stop_time,       // stop time
   double&   buffer           // target array
   ) const
```

Parâmetros

start_time

[in] Início do tempo.

stop_time

[in] Tempo final.

buffer

[in] Referente ao array de destino para os dados

Valor de retorno

>=0 se bem sucedida, -1 em caso de erro.

© 2000-2024, MetaQuotes Ltd.

Classes de Indicador de Tendência

Este grupo de capítulos contém detalhes técnicos das classes de indicador de tendência da Biblioteca Padrão MQL5 e descrições de todos os seus componentes fundamentais.

Classe/grupo	Descrição
CiADX	Average Directional Index
CiADXWilder	Average Directional Index by Welles Wilder
CiBands	Bollinger Bands®
CiEnvelopes	Envelopes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Moving Average
CiSAR	Parabolic Stop And Reverse System
CiStdDev	Standard Deviation
CiDEMA	Double Exponential Moving Average
CiTEMA	Triple Exponential Moving Average
CiFrAMA	Fractal Adaptive Moving Average
CiAMA	Adaptive Moving Average
CiVIDyA	Variable Index Dynamic Average

© 2000-2024, MetaQuotes Ltd.

CiADX

CiADX é uma classe destinada ao uso do indicador técnico Average Directional Index.

Descrição

A classe CiADX possibilita a criação e acesso aos dados do indicador Average Directional Index.

Declaração

```
class CiADX: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiADX

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer da linha principal
Plus	Retorna o elemento de buffer da linha +DI
Minus	Retorna o elemento de buffer da linha -DI
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Plus

Retorna o elemento de buffer da linha +DI pelo índice especificado.

```
double  Plus(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha +DI do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Minus

Retorna o elemento de buffer da linha -DI pelo índice especificado.

```
double  Minus(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha -DI do índice especificado, ou EMPTY_VALUE se não houver qualquer dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_ADX para CiADX).

© 2000-2024, MetaQuotes Ltd.

CiADXWilder

CiADXWilder é uma classe destinada ao uso do indicador técnico Average Directional Index by Welles Wilder.

Descrição

A classe CiADXWilder possibilita a criação e acesso aos dados do indicador Average Directional Index by Welles Wilder.

Declaração

```
class CiADXWilder: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiADXWilder

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer da linha principal
Plus	Retorna o elemento de buffer da linha +DI
Minus	Retorna o elemento de buffer da linha -DI
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Plus

Retorna o elemento de buffer da linha +DI pelo índice especificado.

```
double  Plus(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha +DI do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Minus

Retorna o elemento de buffer da linha -DI pelo índice especificado.

```
double  Minus(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha -DI do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_ADXW para CiADXWilder).

© 2000-2024, MetaQuotes Ltd.

CiBands

CiBands é uma classe destinada ao uso do indicador técnico Bollinger Bands®.

Descrição

A classe CiBands possibilita a criação e o acesso aos dados do indicador Bollinger Bands.

Declaração

```
class CiBands: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBands

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
MaShift	Retorna o deslocamento horizontal
Deviation	Retorna o desvio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Base	Retorna o elemento de buffer da linha base
Upper	Retorna o elemento de buffer da linha superior
Lower	Retorna o elemento de buffer da linha inferior
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retorna o deslocamento horizontal do indicador.

```
int  MaShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Deviation

Retorna o desvio.

```
double  Deviation() const
```

Valor de retorno

Retorna o desvio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Shift
   double           deviation,     // Deviation
   int              applied        // applied price, or handle
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal do indicador.

deviation

[in] Desvio.

applied

[in] Tipo de volume para aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Base

Retorna o elemento de buffer da linha base do índice especificado.

```
double  Base(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha base do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Upper

Retorna o elemento de buffer da linha superior pelo índice especificado.

```
double  Upper(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha superior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Lower

Retorna o elemento de buffer da linha inferior pelo índice especificado.

```
double  Lower(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha inferior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_BANDS para CiBands).

© 2000-2024, MetaQuotes Ltd.

CiEnvelopes

CiEnvelopes é uma classe destinada ao uso do indicador técnico Envelopes.

Descrição

A classe CiEnvelopes possibilita a criação e o acesso aos dados do indicador Envelopes.

Declaração

```
class CiEnvelopes: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiEnvelopes

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
MaShift	Retorna o deslocamento horizontal
MaMethod	Retorna o método de média
Deviation	Retorna o desvio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Upper	Retorna o elemento de buffer da linha superior
Lower	Retorna o elemento de buffer da linha inferior
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retorna o deslocamento horizontal do indicador.

```
int  MaShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Deviation

Retorna o valor do desvio.

```
double  Deviation() const
```

Valor de retorno

Retorna o valor do desvio, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Horizontal shift
   ENUM_MA_METHOD   ma_method,     // Averaging method
   int              applied,       // Price type or handle to apply
   double           deviation      // Deviation
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de Preço do manipulador para aplicar.

deviation

[in] Desvio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Upper

Retorna o elemento de buffer da linha superior pelo índice especificado.

```
double  Upper(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha superior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Lower

Retorna o elemento de buffer da linha inferior pelo índice especificado.

```
double  Lower(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha inferior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_ENVELOPES para CiEnvelopes).

© 2000-2024, MetaQuotes Ltd.

CiIchimoku

CiIchimoku é uma classe destinada ao do indicador técnico Ichimoku Kinko Hyo.

Descrição

A classe CiIchimoku possibilita a criação, setup e acesso aos dados do indicador Ichimoku Kinko Hyo.

Declaração

```
class CiIchimoku: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiIchimoku

Métodos de classe

Atributos	
TenkanSenPeriod	Retorna o período TenkanSen
KijunSenPeriod	Retorna o período KijunSen
SenkouSpanBPeriod	Retorna o período SenkouSpanB
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
TenkanSen	Retorna o elemento de buffer da linha TenkanSen
KijunSen	Retorna o elemento de buffer da linha KijunSen
SenkouSpanA	Retorna o elemento de buffer da linha SenkouSpanA
SenkouSpanB	Retorna o elemento de buffer da linha SenkouSpanB

© 2000-2024, MetaQuotes Ltd.

Atributos	
ChinkouSpan	Retorna o elemento de buffer da linha ChikouSpan
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

TenkanSenPeriod

Retorna o período TenkanSen.

```
int  TenkanSenPeriod() const
```

Valor de retorno

Retorna o período TenkanSen, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

KijunSenPeriod

Retorna o período KijunSen.

```
int  KijunSenPeriod() const
```

Valor de retorno

Retorna o período KijunSen, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SenkouSpanBPeriod

Retorna o período SenkouSpanB.

```
int  SenkouSpanBPeriod() const
```

Valor de retorno

Retorna o período SenkouSpanB, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,            // Symbol
   ENUM_TIMEFRAMES  period,            // Period
   int              tenkan_sen,        // Period of TenkanSen
   int              kijun_sen,         // Period of KijunSen
   int              senkou_span_b      // Period of SenkouSpanB
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

tenkan_sen

[in] Período do TenkanSen.

kijun_sen

[in] Período do KijunSen.

senkou_span_b

[in] Período do SenkouSpanB.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

TenkanSen

Retorna o elemento de buffer da linha TenkanSen pelo índice especificado.

```
double  TenkanSen(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha TenkanSen do índice especificado, ou EMPTY_VALUEse não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

KijunSen

Retorna o elemento de buffer da linha KijunSen pelo índice especificado.

```
double  KijunSen(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha KijunSen de índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

SenkouSpanA

Retorna o elemento de buffer da linha SenkouSpanA pelo índice especificado.

```
double  SenkouSpanA(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento buffer da linha SenkouSpanA do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

SenkouSpanB

Retorna o elemento de buffer da linha SenkouSpanB pelo índice especificado.

```
double  SenkouSpanB(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha SenkouSpanB do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

ChinkouSpan

Retorna o elemento de buffer da linha ChinkouSpan pelo índice especificado.

```
double  ChinkouSpan(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha ChinkouSpan do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_ICHIMOKU para Cilchimoku).

© 2000-2024, MetaQuotes Ltd.

CiMA

CiMA é uma classe destinada ao uso do indicador técnico Moving Average.

Descrição

A classe CiMA possibilita a criação, setup e acesso aos dados do indicador Moving Average.

Declaração

```
class CiMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMA

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
MaShift	Retorna o deslocamento horizontal
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retorna o deslocamento horizontal do indicador.

```
int  MaShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método da média (valor do enumerador ENUM_MA_METHOD), definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           string,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Horizontal shift
   ENUM_MA_METHOD   ma_method,     // Averaging method
   int              applied        // Price type of handle to apply
   )
```

Parâmetros

string

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_MA para CiMA).

© 2000-2024, MetaQuotes Ltd.

CiSAR

CiSAR é uma classe destinada ao uso do indicador técnico Parabolic Stop And Reverse System.

Descrição

A classe CiSAR possibilita a criação, setup e acesso aos dados do indicador Parabolic Stop And Reverse System.

Declaração

```
class CiSAR: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiSAR

Métodos de classe

Atributos	
SarStep	Retorna o passo para o aumento da velocidade
Maximum	Retorna o coeficiente de preço seguinte
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

SarStep

Retorna o passo para o aumento da velocidade (coeficiente de aceleração).

```
double  SarStep() const
```

Valor de retorno

O passo para a velocidade crescente, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Maximum

Retorna o coeficiente do preço seguinte.

```
double  Maximum() const
```

Valor de retorno

O coeficiente do preço seguinte, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,       // Symbol
   ENUM_TIMEFRAMES  period,       // Period
   double           step,         // Step
   double           maximum       // Coefficient
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

step

[in] Passo para velocidade crescente.

maximum

[in] Preço seguindo o coeficiente.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_SAR para CiSAR).

© 2000-2024, MetaQuotes Ltd.

CiStdDev

CiStdDev é uma classe destinada ao uso do indicador técnico Standard Deviation.

Descrição

A classe CiStdDev possibilita a criação, setup e acesso aos dados do indicador Standard Deviation.

Declaração

```
class CiStdDev: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStdDev

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
MaShift	Retorna o deslocamento horizontal
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaShift

Retorna o deslocamento horizontal do indicador.

```
int  MaShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método da média (valor do enumerador ENUM_MA_METHOD), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Horizontal shift
   ENUM_MA_METHOD   ma_method,     // Averaging method
   int              applied        // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_STDDEV para CiStdDev).

© 2000-2024, MetaQuotes Ltd.

CiDEMA

CiDEMA é uma classe destinada ao uso do indicador técnico Double Exponential Moving Average.

Descrição

A classe CiDEMA possibilita a criação, setup e acesso aos dados do indicador Double Exponential Moving Average.

Declaração

```
class CiDEMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDEMA

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
IndShift	Retorna o deslocamento horizontal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retorna o deslocamento horizontal do indicador.

```
int  IndShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           string,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ind_shift,     // Shift
   int              applied        // Price type of handle to apply
   )
```

Parâmetros

string

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ind_shift

[in] Deslocamento horizontal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_DEMA para CiDEMA).

© 2000-2024, MetaQuotes Ltd.

CiTEMA

CiTEMA é uma classe destinada ao uso do indicador técnico Triple Exponential Moving Average.

Descrição

A classe CiTEMA possibilita a criação, setup e acesso aos dados do indicador Triple Exponential Moving Average.

Declaração

```
class CiTEMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiTEMA

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
IndShift	Retorna o deslocamento horizontal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retorna o deslocamento horizontal do indicador.

```
int  IndShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Offset
   int              applied        // Price type of handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_TEMA para CiTEMA).

© 2000-2024, MetaQuotes Ltd.

CiFrAMA

CiFrAMA é uma classe destinada ao uso do indicador técnico Fractal Adaptive Moving Average.

Descrição

A classe CiFrAMA possibilita a criação, setup e acesso aos dados do indicador Fractal Adaptive Moving Average.

Declaração

```
class CiFrAMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiFrAMA

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
IndShift	Retorna o deslocamento horizontal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retorna o deslocamento horizontal do indicador.

```
int  IndShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              ma_shift,      // Offset
   int              applied        // Price type of handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_shift

[in] Deslocamento horizontal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_FRAMA para CiFrAMA).

© 2000-2024, MetaQuotes Ltd.

CiAMA

CiAMA é uma classe destinada ao uso do indicador técnico Adaptive Moving Average.

Descrição

A classe CiAMA possibilita a criação, setup e acesso aos dados do indicador Adaptive Moving Average.

Declaração

```
class CiAMA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAMA

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
FastEmaPeriod	Retorna o período médio da EMA rápida
SlowEmaPeriod	Retorna o período médio da EMA lenta
IndShift	Retorna o deslocamento horizontal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retorna o período médio da EMA rápida.

```
int  FastEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA rápida, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retorna o período médio da EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA lenta, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retorna o deslocamento horizontal do indicador.

```
int  IndShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           string,              // Symbol
   ENUM_TIMEFRAMES  period,              // Period
   int              ma_period,           // Averaging period
   int              fast_ema_period,     // Fast EMA period
   int              slow_ema_period,     // Slow EMA period
   int              ind_shift,           // Shift
   int              applied              // Price type or handle to apply
   )
```

Parâmetros

string

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

fast_ema_period

[in] Período médio da EMA rápida.

slow_ema_period

[in] Período médio da EMA lenta.

ind_shift

[in] Deslocamento horizontal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_AMA para CIAMA).

© 2000-2024, MetaQuotes Ltd.

CiVIDyA

CiVIDyA é uma classe destinada ao suo do indicador técnico Variable Index Dynamic Average.

Descrição

A classe CiVIDyA possibilita a criação, setup e acesso aos dados do indicador Variable Index Dynamic Average.

Declaração

```
class CiVIDyA: public CIndicator
```

Título

```
#include <Indicators\Trend.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVIDyA

Métodos de classe

Atributos	
CmoPeriod	Retorna o período para Momentum
EmaPeriod	Retorna o período médio para EMA
IndShift	Retorna o deslocamento horizontal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

CmoPeriod

Retorna o período para Momentum.

```
int  CmoPeriod() const
```

Valor de retorno

Retorna o período para Momentum, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

EmaPeriod

Retorna o período médio para EMA.

```
int  EmaPeriod() const
```

Valor de retorno

Retorna o período médio para EMA, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

IndShift

Retorna o deslocamento horizontal do indicador.

```
int  IndShift() const
```

Valor de retorno

Retorna o valor de deslocamento horizontal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,         // Symbol
   ENUM_TIMEFRAMES  period,         // Period
   int              cmo_period,     // Momentum period
   int              ema_period,     // Averaging period
   int              ind_shift,      // Shift
   int              applied         // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

cmo_period

[in] Período do momentum.

ema_period

[in] Período médio.

ind_shift

[in] Deslocamento horizontal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_VIDYA para CiVIDyA).

© 2000-2024, MetaQuotes Ltd.

Osciladores

Este grupo de capítulos contém os detalhes técnicos das classes Oscillators da Biblioteca Padrão MQL5 e descrições de todos os seus componentes fundamentais.

Classe/grupo	Descrição
CiATR	Average True Range
CiBearsPower	Bears Power
CiBullsPower	Bulls Power
CiCCI	Commodity Channel Index
CiChaikin	Chaikin Oscillator
CiDeMarker	DeMarker
CiForce	Force Index
CiMACD	Moving Averages Convergence-Divergence
CiMomentum	Momentum
CiOsMA	Moving Average of Oscillator (MACD histogram)
CiRSI	Relative Strength Index
CiRVI	Relative Vigor Index
CiStochastic	Stochastic Oscillator
CiWPR	Williams' Percent Range
CiTriX	Triple Exponential Moving Averages Oscillator

© 2000-2024, MetaQuotes Ltd.

CiATR

CiATR é uma classe destinada ao uso do indicador técnico Average True Range.

Descrição

A classe CiATR possibilita a criação, setup e acesso aos dados do indicador Average True Range.

Declaração

```
class CiATR: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiATR

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_ATR para CiATR).

© 2000-2024, MetaQuotes Ltd.

CiBearsPower

CiBearsPower é uma classe destinada ao uso do indicador técnico Bears Power.

Descrição

A classe CiBearsPower possibilita a criação, setup e acesso aos dados do indicador de Bears Power.

Declaração

```
class CiBearsPower: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBearsPower

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_BEARS para CiBearsPower).

© 2000-2024, MetaQuotes Ltd.

CiBullsPower

CiBullsPower é uma classe destinada ao uso do indicador técnico Bulls Power.

Descrição

A classe CiBullsPower possibilita a criação, setup e acesso aos dados do indicador Bulls Power.

Declaração

```
class CiBullsPower: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBullsPower

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_BULLS para CiBullsPower).

© 2000-2024, MetaQuotes Ltd.

CiCCI

CiCCI é uma classe destinada ao uso do indicador técnico Commodity Channel Index.

Descrição

A classe CiCCI possibilita a criação, setup e acesso aos dados do indicador Commodity Channel Index.

Declaração

```
class CiCCI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiCCI

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              applied        // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_CCI para CiCCi).

© 2000-2024, MetaQuotes Ltd.

CiChaikin

CiChaikin é uma classe destinada ao uso do indicador técnico Chaikin Oscillator.

Descrição

A classe CiChaikin fornece a criação, setup e acesso aos dados do indicador Chaikin Oscillator.

Declaração

```
class CiChaikin: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiChaikin

Métodos de classe

Atributos	
FastMaPeriod	Retorna o período médio para a MA rápida
SlowMaPeriod	Retorna o período médio para a MA lenta
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastMaPeriod

Retorna o período médio da EMA rápida.

```
int  FastMaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA rápida, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SlowMaPeriod

Retorna o período médio da EMA lenta.

```
int  SlowMaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA lenta, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string                symbol,              // Symbol
   ENUM_TIMEFRAMES       period,              // Period
   int                   fast_ma_period,      // Fast EMA period
   int                   slow_ma_period,      // Slow EMA period
   ENUM_MA_METHOD        ma_method,           // Averaging method
   ENUM_APPLIED_VOLUME   applied              // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

fast_ma_period

[in] Período da EMA rápida.

slow_ma_period

[in] Período da EMA lenta.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de volume para aplicar (enumeração ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_CHAIKIN para CiChaikin).

© 2000-2024, MetaQuotes Ltd.

CiDeMarker

CiDeMarker é uma classe destinada ao uso do indicador técnico DeMarker.

Descrição

A classe CiDeMarker fornece a criação, setup e acesso aos dados do indicador DeMarker.

Declaração

```
class CiDeMarker: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDeMarker

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_DEMARKER para CiDeMarker).

© 2000-2024, MetaQuotes Ltd.

CiForce

CiForce é uma classe destinada ao uso do indicador técnico Force Index.

Descrição

A classe CiForce fornece a criação, setup e acesso aos dados do indicador Force Index.

Declaração

```
class CiForce: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiForce

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string                symbol,         // Symbol
   ENUM_TIMEFRAMES       period,         // Period
   int                   ma_period,      // Averaging period
   ENUM_MA_METHOD        ma_method,      // Averaging method
   ENUM_APPLIED_VOLUME   applied         // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de volume para aplicar (enumeração ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_FORCE para CiForce).

© 2000-2024, MetaQuotes Ltd.

CiMACD

CiMACD é uma classe destinada ao uso do indicador técnico Moving Averages Convergence-Divergence.

Descrição

A classe CiMACD fornece a criação, setup e acesso aos dados do indicador Moving Averages Convergence-Divergence.

Declaração

```
class CiMACD: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMACD

Métodos de classe

Atributos	
FastEmaPeriod	Retorna o período médio da EMA rápida
SlowEmaPeriod	Retorna o período médio da EMA lenta
SignalPeriod	Retorna o período médio da linha de sinal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer da linha principal
Signal	Retorna o elemento de buffer da linha de sinal
Entrada/saída	

© 2000-2024, MetaQuotes Ltd.

Atributos	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retorna o período médio da EMA rápida.

```
int  FastEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA rápida, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retorna o período médio da EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA lenta, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SignalPeriod

Retorna o período médio da linha de sinal.

```
int  SignalPeriod() const
```

Valor de retorno

Retorna o período médio da linha de sinal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,              // Symbol
   ENUM_TIMEFRAMES  period,              // Period
   int              fast_ema_period,     // Fast EMA period
   int              slow_ema_period,     // Slow EMA period
   int              signal_period,       // Signal period
   int              applied              // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

fast_ema_period

[in] Período da EMA rápida.

slow_ema_period

[in] Período da EMA lenta.

signal_period

[in] Período da linha de sinal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Signal

Retorna o elemento de buffer da linha de sinal pelo índice especificado.

```
double  Signal(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha de sinal do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_MACD para CiMACD).

© 2000-2024, MetaQuotes Ltd.

CiMomentum

CiMomentum é uma classe destinada ao uso do indicador técnico Momentum.

Descrição

A classe CiMomentum fornece a criação, setup e acesso aos dados do indicador Momentum.

Declaração

```
class CiMomentum: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              applied        // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_MOMENTUM para CiMomentum).

© 2000-2024, MetaQuotes Ltd.

CiOsMA

CiOsMA é uma classe destinada ao uso do indicador técnico Moving Average of Oscillator (histograma MACD).

Descrição

A classe CiOsMA fornece a criação, setup e acesso aos dados do indicador Moving Average of Oscillator (histograma MACD).

Declaração

```
class CiOsMA: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOsMA

Métodos de classe

Atributos	
FastEmaPeriod	Retorna o período médio da EMA rápida
SlowEmaPeriod	Retorna o período médio da EMA lenta
SignalPeriod	Retorna o período médio da linha de sinal
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

FastEmaPeriod

Retorna o período médio da EMA rápida.

```
int  FastEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA rápida, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SlowEmaPeriod

Retorna o período médio da EMA lenta.

```
int  SlowEmaPeriod() const
```

Valor de retorno

Retorna o período médio da EMA lenta, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

SignalPeriod

Retorna o período médio da linha de sinal.

```
int  SignalPeriod() const
```

Valor de retorno

Retorna o período médio da linha de sinal, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,              // Symbol
   ENUM_TIMEFRAMES  period,              // Period
   int              fast_ema_period,     // Fast EMA period
   int              slow_ema_period,     // Slow EMA period
   int              signal_period,       // Signal line period
   int              applied              // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

fast_ema_period

[in] Período da EMA rápida.

slow_ema_period

[in] Período da EMA lenta.

signal_period

[in] Período da linha de sinal.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_OSMA para CiOsMA).

© 2000-2024, MetaQuotes Ltd.

CiRSI

CiRSI é uma classe destinada ao uso do indicador técnico Relative Strength Index.

Descrição

A classe CiRSI fornece a criação, setup e acesso aos dados do indicador Relative Strength Index.

Declaração

```
class CiRSI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiRSI

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              applied        // Price type or handle to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

applied

[in] Tipo de preço ou manipulador a aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_RSI para CiRSI).

© 2000-2024, MetaQuotes Ltd.

CiRVI

CiRVI é uma classe destinada ao uso do indicador técnico Relative Vigor Index.

Descrição

A classe CiRVI fornece a criação, setup e acesso aos dados do indicador Relative Vigor Index.

Declaração

```
class CiRVI: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiRVI

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer da linha base
Signal	Retorna o elemento de buffer da linha de sinal
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period      // Averaging period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Signal

Retorna o elemento de buffer da linha de sinal pelo índice especificado.

```
double  Signal(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha de sinal do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto do identificador (IND_RVIpara CiRVI).

© 2000-2024, MetaQuotes Ltd.

CiStochastic

CiStochastic é uma classe destinada ao uso do indicador técnico Stochastic Oscillator.

Descrição

A classe CiStochastic fornece a criação, setup e acesso aos dados do indicador Stochastic Oscillator.

Declaração

```
class CiStochastic: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStochastic

Métodos de classe

Atributos	
Kperiod	Retorna o período médio para a linha %K
Dperiod	Retorna o período médio para a linha %D
Slowing	Retorna o período de desaceleração
MaMethod	Retorna o método de média
PriceField	Tipo de Preço (Low/High ou Close/Close) para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer da linha base
Signal	Retorna o elemento de buffer da linha de sinal
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Kperiod

Retorna o período médio da linha %K.

```
int  Kperiod() const
```

Valor de retorno

Retorna o período médio da linha %K, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Dperiod

Retorna o período médio da linha %D.

```
int  Dperiod() const
```

Valor de retorno

Retorna o período médio da linha %D, definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

Slowing

Retorna o período de desaceleração.

```
int  Slowing() const
```

Valor de retorno

Retorna o período de desaceleração, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

PriceField

Retorna o tipo de preço (Low/High ou Close/Close) para aplicar.

```
ENUM_STO_PRICE  PriceField() const
```

Valor de retorno

O tipo de preço para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,          // Symbol
   ENUM_TIMEFRAMES  period,          // Period
   int              Kperiod,         // Averaging period of %K
   int              Dperiod,         // Averaging period of %D
   int              slowing,         // Slowing period
   ENUM_MA_METHOD   ma_method,       // Averaging method
   ENUM_STO_PRICE   price_field      // Price type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Kperiod

[in] Período médio da linha %K.

Dperiod

[in] Período médio da linha %D.

slowing

[in] Período de desaceleração.

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

price_field

[in] Tipo de Preço (Low/High ou Close/Close) para aplicar (enumerador ENUM_STO_PRICE).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Signal

Retorna o elemento de buffer da linha de sinal pelo índice especificado.

```
double  Signal(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Signal

Retorna o elemento de buffer da linha de sinal pelo índice especificado.

```
double  Signal(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha de sinal do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto do identificador (IND_STOCHASTIC para CiStochastic).

© 2000-2024, MetaQuotes Ltd.

CiTriX

CiTriX é uma classe destinada ao uso do indicador técnico Triple Exponential Moving Averages Oscillator.

Descrição

A classe CiTriX prevê a criação, setup e acesso aos dados do indicador Triple Exponential Moving Averages Oscillator.

Declaração

```
class CiTriX: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiTriX

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,        // Symbol
   ENUM_TIMEFRAMES  period,        // Period
   int              ma_period,     // Averaging period
   int              applied        // Price type or handle
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

applied

[in] Tipo de Preço do manipulador para aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_TRIX para Citrix).

© 2000-2024, MetaQuotes Ltd.

CiWPR

CiWPR é uma classe destinada ao uso do indicador técnico Williams' Percent Range.

Descrição

A classe CiWPR fornece a criação, setup e acesso aos dados do indicador Williams' Percent Range.

Declaração

```
class CiWPR: public CIndicator
```

Título

```
#include <Indicators\Oscilators.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiWPR

Métodos de classe

Atributos	
CalcPeriod	Retorna o período calculado
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

CalcPeriod

Retorna o período calculado.

```
int  CalcPeriod() const
```

Valor de retorno

Retorna o período calculado, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,          // Symbol
   ENUM_TIMEFRAMES  period,          // Period
   int              calc_period      // Calculation period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

calc_period

[in] Período para cálculo.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_WPR para CiWPR).

© 2000-2024, MetaQuotes Ltd.

Indicadores de Volume

Este grupo de capítulos contém detalhes técnicos das classes dos Indicadores de Volume da Biblioteca Padrão MQL5 e descrições de todos os seus componentes fundamentais.

Classe/grupo	Descrição
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

© 2000-2024, MetaQuotes Ltd.

CiAD

CiAD é uma classe destinada ao uso do indicador técnico Accumulation/Distribution.

Descrição

A classe CiAD possibilita a criação, setup e acesso aos dados do indicador Accumulation/Distribution.

Declaração

```
class CiAD: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAD

Métodos de classe

Atributos	
Applied	Retorna o tipo de volume para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string               symbol,      // Symbol
   ENUM_TIMEFRAMES      period,      // Period
   ENUM_APPLIED_VOLUME  applied      // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

applied

[in] Tipo de volume para aplicar (enumerador ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_AD para CiAD).

© 2000-2024, MetaQuotes Ltd.

CiMFI

CiMFI é uma classe destinada ao uso do indicador técnico Money Flow Index.

Descrição

A classe CiMFI possibilita a criação, setup e acesso aos dados do indicador Money Flow Index.

Declaração

```
class CiMFI: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMFI

Métodos de classe

Atributos	
MaPeriod	Retorna o período médio
Applied	Retorna o tipo de volume para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

MaPeriod

Retorna o período médio.

```
int  MaPeriod() const
```

Valor de retorno

Retorna o período médio, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string               symbol,         // Symbol
   ENUM_TIMEFRAMES      period,         // Period
   int                  ma_period,      // Averaging period
   ENUM_APPLIED_VOLUME  applied         // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

ma_period

[in] Período médio.

applied

[in] Tipo de volume para aplicar (enumerador ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_MFI para CiMFI).

© 2000-2024, MetaQuotes Ltd.

CiOBV

CiOBV é uma classe destinada ao uso do indicador técnico On Balance Volume.

Descrição

A classe CiOBV possibilita a criação, setupo e acesso aos dados do indicador On Balance Volume.

Declaração

```
class CiOBV: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiOBV

Métodos de classe

Atributos	
Applied	Retorna o tipo de volume para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string               symbol,      // Symbol
   ENUM_TIMEFRAMES      period,      // Period
   ENUM_APPLIED_VOLUME  applied      // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

applied

[in] Tipo de volume para aplicar (enumerador ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_OBV para CiOBV).

© 2000-2024, MetaQuotes Ltd.

CiVolumes

CiVolumes é uma classe destinada ao uso do indicador técnico Volumes.

Descrição

A classe CiVolumes posibilita a criação, setup e acesso aos dados do indicador Volumes.

Declaração

```
class CiVolumes: public CIndicator
```

Título

```
#include <Indicators\Volumes.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVolumes

Métodos de classe

Atributos	
Applied	Retorna o tipo de volume para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string               symbol,      // Symbol
   ENUM_TIMEFRAMES      period,      // Period
   ENUM_APPLIED_VOLUME  applied      // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

applied

[in] Tipo de volume para aplicar (enumerador ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_VOLUMES para CiVolumes).

© 2000-2024, MetaQuotes Ltd.

Indicadores Bill Williams

Este grupo de capítulos contém detalhes técnicos das classes do indicador Bill Williams da Biblioteca Padrão MQL5 e descrições de todos os seus componentes fundamentais.

Classe/grupo	Descrição
CiAC	Accelerator Oscillator
CiAlligator	Alligator
CiAO	Awesome Oscillator
CiFractals	Fractals
CiGator	Gator Oscillator
CiBWMFI	Market Facilitation Index

© 2000-2024, MetaQuotes Ltd.

CiAC

CIAC é uma classe do indicador técnico Oscillator Accelerator.

Descrição

A classe CIAC fornece a criação, setup e acesso aos dados do indicador Accelerator oscilador.

Declaração

```
class CiAC: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAC

Métodos de classe

Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,      // Symbol
   ENUM_TIMEFRAMES  period       // Period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Objeto tipo identificado (IND_AC para CiAC).

© 2000-2024, MetaQuotes Ltd.

CiAlligator

CiAlligator é uma classe do indicador técnico Alligator.

Descrição

A classe CiAlligator fornece a criação, setup e acesso aos dados do indicador Alligator.

Declaração

```
class CiAlligator: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAlligator

Métodos de classe

Atributos	
JawPeriod	Retorna o período médio para a linha das mandíbulas (Jaws)
JawShift	Retorna o deslocamento horizontal da linha das mandíbulas (Jaws)
TeethPeriod	Retorna o período médio para a linha dos dentes (Teeths)
TeethShift	Retorna o deslocamento horizontal da linha dos dentes (Teeths)
LipsPeriod	Retorna o período médio para a linha dos lábios (Lips)
LipsShift	Retorna o deslocamento horizontal da linha dos lábios (Lips)
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Jaw	Retorna o elemento de buffer da linha das mandíbulas (Jaws)

© 2000-2024, MetaQuotes Ltd.

Atributos	
Teeth	Retorna o elemento de buffer da linha dos dentes (Teeths)
Lips	Retorna o elemento de buffer da linha dos lábios (Lips)
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

JawPeriod

Retorna o período médio para a linha da mandíbula (Jaw)

```
int  JawPeriod() const
```

Valor de retorno

Retorna o período médio para a linha da mandíbula (Jaw), definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

JawShift

Retorna o deslocamento horizontal da linha das mandíbulas (Jaws).

```
int  JawShift() const
```

Valor de retorno

Deslocamento horizontal da linha das mandíbilas (Jaws), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

TeethPeriod

Retorna o período médio para a linha dos dentes (Teeth).

```
int  TeethPeriod() const
```

Valor de retorno

Retorna o período médio para a linha dos dentes (Teeth), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

TeethShift

Retorna o deslocamento horizontal da linha dos dentes (Teeths).

```
int  TeethShift() const
```

Valor de retorno

Deslocamento horizontal da linha dos dentes (Teeths), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

LipsPeriod

Retorna o período médio para a linha dos lábios (Lips).

```
int  LipsPeriod() const
```

Valor de retorno

Retorna o período médio para a linha dos lábios (Lips), definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

LipsShift

Retorna o deslocamento horizontal da linha dos lábios (Lips).

```
int  LipsShift() const
```

Valor de retorno

Deslocamento horizontal da linha dos lábios (Lips), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,           // Symbol
   ENUM_TIMEFRAMES  period,           // Period
   int              jaw_period,       // Jaws period
   int              jaw_shift,        // Jaws shift
   int              teeth_period,     // Teeths period
   int              teeth_shift,      // Teeths shift
   int              lips_period,      // Lips period
   int              lips_shift,       // Lips shift
   ENUM_MA_METHOD   ma_method,        // Averaging method
   int              applied           // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

jaw_period

[in] Período das mandíbulas (Jaws).

jaw_shift

[in] Deslocamento das mandíbulas (Jaws).

teeth_period

[in] Período dos dentes (Teeths).

teeth_shift

[in] Deslocamento dos dentes (Teeths).

lips_period

[in] Período dos lábios (Lips)

lips_shift

[in] Deslocamento dos lábios (Lips).

ma_method

[in] Método da média móvel (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de volume para aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Jaw

Retorna o elemento de buffer da linha das mandíbulas (Jaws) pelo índice especificado.

```
double  Jaw(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha das mandíbulas (Jaws) do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Teeth

Retorna o elemento de buffer da linha dos dentes (Teeths) pelo índice especificado.

```
double  Teeth(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha dos dentes (Teeths) do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Lips

Retorna o elemento de buffer da linha dos lábios (Lips) pelo índice especificado.

```
double  Lips(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer da linha dos lábios (Lips) do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Objeto do tipo identificador (IND_ALLIGATOR para CiAlligator).

© 2000-2024, MetaQuotes Ltd.

CiAO

CiAO é uma classe do indicador técnico Awesome Oscillator.

Descrição

A classe CiAO fornece a criação, setup e acesso aos dados do indicador Awesome Oscillator.

Declaração

```
class CiAO: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiAO

Métodos de classe

Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,      // Symbol
   ENUM_TIMEFRAMES  period       // Period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_AO para CiAO).

© 2000-2024, MetaQuotes Ltd.

CiFractals

CiFractals é uma classe do indicador técnico Fractals.

Descrição

A classe CiFractals fornece a criação, setup e acesso aos dados do indicador Fractals.

Declaração

```
class CiFractals: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiFractals

Métodos de classe

Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Upper	Retorna o elemento de buffer superior
Lower	Retorna o elemento de buffer inferior
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,      // Symbol
   ENUM_TIMEFRAMES  period       // Period
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Upper

Retorna o elemento de buffer referente ao buffer superior pelo índice especificado.

```
double  Upper(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer referente ao buffer superior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Lower

Retorna o elemento de buffer referente ao buffer inferior pelo índice especificado.

```
double  Lower(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer referente ao buffer inferior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_FRACTALS para CiFractals).

© 2000-2024, MetaQuotes Ltd.

CiGator

CiGator é uma classe do indicador técnico Gator Oscilador.

Descrição

A classe CiGator fornece a criação, setup e acesso aos dados do indicador Gator Oscilador.

Declaração

```
class CiGator: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiGator

Métodos de classe

Atributos	
JawPeriod	Retorna o período médio para a linha das mandíbulas (Jaws)
JawShift	Retorna o deslocamento horizontal da linha das mandíbulas (Jaws)
TeethPeriod	Retorna o período médio para a linha dos dentes (Teeths)
TeethShift	Retorna o deslocamento horizontal da linha dos dentes (Teeths)
LipsPeriod	Retorna o período médio para a linha dos lábios (Lips)
LipsShift	Retorna o deslocamento horizontal da linha dos lábios (Lips)
MaMethod	Retorna o método de média
Applied	Retorna o tipo de preço ou manipulador para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Upper	Retorna o elemento de buffer superior

© 2000-2024, MetaQuotes Ltd.

Atributos	
Lower	Retorna o elemento de buffer inferior
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

JawPeriod

Retorna o período médio para a linha das Mandíbulas (Jaws).

```
int  JawPeriod() const
```

Valor de retorno

Retorna o período médio para a linha das Mandíbulas (Jaws), definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

JawShift

Retorna o deslocamento horizontal da linha das mandíbulas (Jaws).

```
int  JawShift() const
```

Valor de retorno

Deslocamento horizontal da linha das mandíbilas (Jaws), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

TeethPeriod

Retorna o período médio para a linha dos dentes (Teeth).

```
int  TeethPeriod() const
```

Valor de retorno

Retorna o período médio para a linha dos dentes (Teeth), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

TeethShift

Retorna o deslocamento horizontal da linha dos dentes (Teeths).

```
int  TeethShift() const
```

Valor de retorno

Deslocamento horizontal da linha dos dentes (Teeths), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

LipsPeriod

Retorna o período médio para a linha dos lábios (Lips).

```
int  LipsPeriod() const
```

Valor de retorno

Retorna o período médio para a linha dos lábios (Lips), definido na criação indicador.

© 2000-2024, MetaQuotes Ltd.

LipsShift

Retorna o deslocamento horizontal da linha dos lábios (Lips).

```
int  LipsShift() const
```

Valor de retorno

Deslocamento horizontal da linha dos lábios (Lips), definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

MaMethod

Retorna o método da média.

```
ENUM_MA_METHOD  MaMethod() const
```

Valor de retorno

Retorna o método de média, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de preço ou manipulador para aplicar.

```
int  Applied() const
```

Valor de retorno

Tipo de preço ou manipulador para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string           symbol,           // Symbol
   ENUM_TIMEFRAMES  period,           // Period
   int              jaw_period,       // Jaws period
   int              jaw_shift,        // Jaws shift
   int              teeth_period,     // Teeths period
   int              teeth_shift,      // Teeths shift
   int              lips_period,      // Lips period
   int              lips_shift,       // Lips shift
   ENUM_MA_METHOD   ma_method,        // Averaging method
   int              applied           // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

jaw_period

[in] Período das mandíbulas (Jaws).

jaw_shift

[in] Deslocamento das mandíbulas (Jaws).

teeth_period

[in] Período dos dentes (Teeths).

teeth_shift

[in] Deslocamento dos dentes (Teeths).

lips_period

[in] Período dos lábios (Lips)

lips_shift

[in] Deslocamento dos lábios (Lips).

ma_method

[in] Método de média (enumerador ENUM_MA_METHOD).

applied

[in] Tipo de volume para aplicar.

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Upper

Retorna o elemento de buffer referente ao buffer superior pelo índice especificado.

```
double  Upper(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer referente ao buffer superior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Lower

Retorna o elemento de buffer referente ao buffer inferior pelo índice especificado.

```
double  Upper(
   int  index       // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

O elemento de buffer referente ao buffer inferior do índice especificado, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_GATOR para CiGator).

© 2000-2024, MetaQuotes Ltd.

CiBWMFI

CiBWMFI é uma classe do Índice de Facilitação do Mercado pelo indicador técnico Bill Williams.

Descrição

A classe CiBWMFI fornece a criação, setup e acesso aos dados do Índice de Facilitação do Mercado pelo indicador técnico Bill Williams.

Declaração

```
class CiBWMFI: public CIndicator
```

Título

```
#include <Indicators\BillWilliams.mqh>
```

Hierarquia de herança

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBWMFI

Métodos de classe

Atributos	
Applied	Retorna o tipo de volume para aplicar
Métodos de Criação	
Create	Cria o indicador
Métodos de Acesso aos Dados	
Main	Retorna o elemento de buffer
Entrada/saída	
virtual Type	Retorna o identificador do tipo de objeto

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Compare

Métodos herdados da classe CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Métodos herdados da classe CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Métodos herdados da classe CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2024, MetaQuotes Ltd.

Applied

Retorna o tipo de volume para aplicar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Valor de retorno

Tipo de Volume para aplicar, definido na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o indicador com parâmetros específicos. Use Refresh() e GetData() para atualizar e obter os valores do indicador.

```
bool  Create(
   string               symbol,      // Symbol
   ENUM_TIMEFRAMES      period,      // Period
   ENUM_APPLIED_VOLUME  applied      // Volume type to apply
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

applied

[in] Tipo de volume para aplicar (enumeração ENUM_APPLIED_VOLUME).

Valor de retorno

verdadeiro se teve sucesso, falso se o indicador não foi criado.

© 2000-2024, MetaQuotes Ltd.

Main

Retorna o elemento de buffer pelo índice especificado

```
double  Main(
   int  index      // Index
   )
```

Parâmetros

index

[in] Índice do elemento.

Valor de retorno

Elemento de buffer do índice especificado se tiver sucesso, ou EMPTY_VALUE se não houver nenhum dado correto.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto tipo "identifier".

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador(IND_BWMFI para CiBWMFI).

© 2000-2024, MetaQuotes Ltd.

CiCustom

CiCustom é uma classe destinada ao uso dos indicadores técnicos personalizados.

Descrição

A classe CiCustom fornece a criação, setup e acesso aos dados do indicador personalizado.

Declaração

```
class CiCustom: public CIndicator
```

Título

```
#include <Indicators\Custom.mqh>
```

Métodos de classe

Atributos	
NumBuffers	Define o número de buffers
NumParams	Obtém o número de parâmetros
ParamType	Obtém o tipo do parâmetro especificado
ParamLong	Obtém o valor do parâmetro especificado do tipo integer
ParamDouble	Obtém o valor do parâmetro especificado do tipo double
ParamString	Obtém o valor do parâmetro especificado do tipo string
Entrada/saída	
virtual Type	Obtém o identificador do tipo de objeto

© 2000-2024, MetaQuotes Ltd.

NumBuffers

Define o número de buffers.

```
bool  NumBuffers(
   int  buffers      // number of buffers
   )
```

Valor de retorno

verdadeiro se teve sucesso, falso se os buffers não foram definidas.

© 2000-2024, MetaQuotes Ltd.

NumParams

Obtém o número de parâmetros.

```
int  NumParams() const
```

Valor de retorno

Número de parâmetros, utilizados na criação do indicador.

© 2000-2024, MetaQuotes Ltd.

ParamType

Obtém um tipo de parâmetro com o índice especificado.

```
ENUM_DATATYPE  ParamType(
   int  index      // parameter index
   ) const
```

Parâmetros

index

[in] Índice de parâmetros.

Valor de retorno

Retorna o tipo de dados (valor do enumerador ENUM_DATATYPE) do parâmetro com o índice especificado, utilizado na criação do indicador.

Observação

Se o índice do parâmetro é inválido, retorna WRONG_VALUE.

© 2000-2024, MetaQuotes Ltd.

ParamLong

Obtém o valor do parâmetro especificado do tipo long.

```
long  ParamLong(
   int  index      // index
   ) const
```

Parâmetros

index

[in] Índice de parâmetros.

Valor de retorno

O valor do parâmetro especificado do tipo long, utilizada na criação do indicador.

Observação

Se o número for inválido, ele retorna 0.

© 2000-2024, MetaQuotes Ltd.

ParamDouble

Obtém o valor do parâmetro especificado do tipo double.

```
double  ParamDouble(
   int  index      // index
   ) const
```

Parâmetros

index

[in] Índice de parâmetros.

Valor de retorno

O valor do parâmetro especificado do tipo double, usado na criação do indicador

Observação

Se o número for inválido, ele retorna EMPTY_VALUE.

© 2000-2024, MetaQuotes Ltd.

ParamString

Obtém o valor do parâmetro especificado do tipo string.

```
string  ParamString(
   int  index      // index
   ) const
```

Parâmetros

index

[in] Índice de parâmetros.

Valor de retorno

O valor do parâmetro especificado do tipo string, utilizado para a criação do indicador.

Observação

Se o número for inválido, retorna uma string vazia.

© 2000-2024, MetaQuotes Ltd.

Type

Retorna o objeto do tipo identifier.

```
virtual int  Type() const
```

Valor de retorno

Tipo de objeto identificador (IND_CUSTOM para CiCustom).

© 2000-2024, MetaQuotes Ltd.

Classes de negociação

Esta seção contém detalhes técnicos de como trabalhar com classes de negociação e descrição dos componentes principais da biblioteca padrão MQL5.

Usando as classes de negociação será economizado tempo ao criar programas personalizados (Expert Advisors).

A Biblioteca Padrão MQL5 (em termos de conjuntos de dados) é alocado no diretório de trabalho do terminal, na pasta Include\Arrays.

Class/Group	Descrição
CAccountInfo	Classe para trabalhar com propriedades da conta de negociação
CSymbolInfo	Classe para trabalhar com propriedades de instrumentos de negociação
COrderInfo	Classe para trabalhar com propriedades de ordens pendentes
CHistoryOrderInfo	Classe para trabalhar com propriedades do histórico das ordens
CPositionInfo	Classe para trabalhar com propriedades da abertura de posição
CDealInfo	Classe para trabalhar com propriedades do histórico de ofertas
CTrade	Classe para a execução de operações de negociação
CTerminalInfo	Classe para obter as propriedades de ambiente do programa MQL5

© 2000-2024, MetaQuotes Ltd.

CAccountInfo

CAccountInfo é uma classe para facilitar o acesso às propriedades da conta de negociação atualmente abertas.

Descrição

CAccountInfo classe fornece acesso fácil às propriedades da conta de negociação atualmente abertas.

Declaração

```
class CAccountInfo : public CObject
```

Título

```
#include <Trade\AccountInfo.mqh>
```

Hierarquia de herança

CObject
  CAccountInfo

Métodos de classe pelos grupos

O acesso a propriedades do tipo integer	
Login	Obtém o número de conta
TradeMode	Obtém o modo de negociação
TradeModeDescription	Obtém o modo de negociação como uma string
Leverage	Obtém a quantidade de alavancagem a disposição
StopoutMode	Obtém o modo "stop out" da conta
StopoutModeDescription	Obtém a descrição modo "stop out" da conta
TradeAllowed	Obtém o flag de retorno da negociação
TradeExpert	Obtém o flag de retorno da negociação automatizada
LimitOrders	Obtém o número máximo das ordens pendentes permitidas
MarginMode	Obtém o modo para o cálculo da margem
MarginModeDescription	Obtém o modo de cálculo para a margem como uma linha
O acesso a propriedades do tipo double	
Balance	Obtém o saldo da conta
Credit	Obtém a quantidade de crédito recebido
Profit	Obtém a quantidade de lucro corrente na conta

© 2000-2024, MetaQuotes Ltd.

O acesso a propriedades do tipo integer	
Equity	Obtém a quantidade atual de capital líquido na conta
Margin	Obtém o valor da margem reservada
FreeMargin	Obtém a quantidade de margem livre
MarginLevel	Obtém o nível de margem
MarginCall	Obtém o nível de margem para depósito
MarginStopOut	Obtém o nível de margem para Stop Out
O acesso a propriedades de texto	
Name	Obtém o nome do cliente
Servidor	Obtém o nome do servidor de negociação
Currency	Obtém o nome da moeda de depósito
Company	Obtém o nome da empresa, que serve uma conta
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém valor específico da propriedade tipo string
Métodos adicionais	
OrderProfitCheck	Obtém o lucro avaliado, com base nos parâmetros transmitidos
MarginCheck	Obtém a quantidade de margem que é necessária para executar operação de negociação
FreeMarginCheck	Obtém a quantidade de margem livre remanescente após a execução da operação de negociação
MaxLotCheck	Obtém o volume máximo possível da operação de negociação

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Login

Obtém o número da conta.

```
long  Login() const
```

Valor de retorno

Número da conta.

© 2000-2024, MetaQuotes Ltd.

TradeMode

Obtém o modo de negociação.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Valor de retorno

Modo de negociação (valor do enumerador ENUM_ACCOUNT_TRADE_MODE).

© 2000-2024, MetaQuotes Ltd.

TradeModeDescription

Obtém o modo de negociação como uma string.

```
string  TradeModeDescription() const
```

Valor de retorno

Modo de negociação como uma string.

© 2000-2024, MetaQuotes Ltd.

Leverage

Obtém a quantidade de alavancagem a disposição.

```
long  Leverage() const
```

Valor de retorno

Quantidade de alavancagem a disposição.

© 2000-2024, MetaQuotes Ltd.

StopoutMode

Obtém o modo Stop Out (saída forçada) da conta .

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Valor de retorno

O modo Stop Out da Conta (Valor do enumerador ENUM_ACCOUNT_STOPOUT_MODE).

© 2000-2024, MetaQuotes Ltd.

StopoutModeDescription

Obtém o modo de definir o nível da margem mínima como uma string.

```
string  StopoutModeDescription() const
```

Valor de retorno

Modo de configuração do nível da margem mínima como uma string.

© 2000-2024, MetaQuotes Ltd.

MarginMode

Obtém o modo para o cálculo da margem.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

valor de retorno

Modo de cálculo para a margem a partir da enumeração ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2024, MetaQuotes Ltd.

MarginModeDescription

Obtém o modo de cálculo para a margem como uma linha.

```
string  MarginModeDescription() const
```

valor de retorno

Modo de cálculo para a margem como uma linha.

© 2000-2024, MetaQuotes Ltd.

TradeAllowed

Obtém o flag de retorno da negociação.

```
bool  TradeAllowed() const
```

Valor de retorno

Flag de retorno da negociação.

© 2000-2024, MetaQuotes Ltd.

TradeExpert

Obtém o flag de retorno da negociação automatizada.

```
bool TradeExpert() const
```

Valor de retorno

Flag de retorno da negociação automatizada.

© 2000-2024, MetaQuotes Ltd.

LimitOrders

Obtém o número máximo das ordens pendentes permitidas

```
int  LimitOrders() const
```

Valor de retorno

O número máximo das ordens pendentes permitidas.

Observação

0 - sem limites.

© 2000-2024, MetaQuotes Ltd.

Balance

Obtém o saldo da conta.

```
double  Balance() const
```

Valor de retorno

O saldo da conta (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

Credit

Obtém a quantidade de crédito a disposição.

```
double  Credit() const
```

Valor de retorno

Quantidade de crédito a disposição (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

Profit

Obtém o montante do lucro atualizado na conta.

```
double  Profit() const
```

Valor de retorno

Montante do lucro atualizado na conta (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

Equity

Obtém a quantidade atual de capital líquido na conta.

```
double  Equity() const
```

Valor de retorno

Quantidade atual de capital líquido na conta (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

Margin

Obtém o valor da margem reservada.

```
double  Margin() const
```

Valor de retorno

Valor da margem reservada (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

FreeMargin

Obtém a quantidade de margem livre.

```
double  FreeMargin() const
```

Valor de retorno

Quantidade de margem livre (em moeda de depósito).

© 2000-2024, MetaQuotes Ltd.

MarginLevel

Obtém o nível de margem.

```
double  MarginLevel() const
```

Valor de retorno

Nível de margem.

© 2000-2024, MetaQuotes Ltd.

MarginCall

Obtém o nível de margem para um depósito.

```
double  MarginCall() const
```

Valor de retorno

Nível de margem para um depósito.

© 2000-2024, MetaQuotes Ltd.

MarginStopOut

Obtém o nível de margem para Stop Out (saída forçada).

```
double  MarginStopOut() const
```

Valor de retorno

Nível de margem para Stop Out.

© 2000-2024, MetaQuotes Ltd.

Name

Obtém o nome do cliente.

```
string  Name() const
```

Valor de retorno

Nome do cliente.

© 2000-2024, MetaQuotes Ltd.

Servidor

Obtém o nome do servidor de negociação.

```
string  Server() const
```

Valor de retorno

Nome do servidor de negociação.

© 2000-2024, MetaQuotes Ltd.

Currency

Obtém o nome da moeda de depósito.

```
string  Currency() const
```

Valor de retorno

Nome da moeda de depósito.

© 2000-2024, MetaQuotes Ltd.

Company

Obtém o nome da empresa, que serve uma conta.

```
string  Company() const
```

Valor de retorno

Nome da empresa, que serve uma conta.

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // property ID
   ) const
```

Parâmetros

prop_id

[in] Identificador da propriedade. O valor pode ser um dos valores do enumerador ENUM_ACCOUNT_INFO_INTEGER.

Valor de retorno

Valor de tipo long.

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // property ID
   ) const
```

Parâmetros

prop_id

[in] Identificador da propriedade. O valor pode ser um dos valores do enumerador ENUM_ACCOUNT_INFO_DOUBLE.

Valor de retorno

Valor de tipo double.

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // property ID
   ) const
```

Parâmetros

prop_id

[in] Identificador da propriedade. O valor pode ser um dos valores do enumerador ENUM_ACCOUNT_INFO_STRING.

Valor de retorno

Valor de tipo string.

© 2000-2024, MetaQuotes Ltd.

OrderProfitCheck

A função calcula o lucro para a conta corrente, com base nos parâmetros transmitidos. A função é usada para pré-avaliação do resultado de uma operação de negociação. O valor é retornado na moeda da conta.

```
double  OrderProfitCheck(
   const string        symbol,              // symbol
   ENUM_ORDER_TYPE     trade_operation,     // operation type (ORDER_TYPE_BUY or ORDE
   double              volume,              // volume
   double              price_open,          // open price
   double              price_close          // close price
   ) const
```

Parâmetros

symbol

[in] Símbolo para a operação de negociação.

trade_operation

[in] Tipo de operação de negociação (enumerador ENUM_ORDER_TYPE).

volume

[in] Volume de operação de negociação.

price_open

[in] Preço de abertura.

price_close

[in] Preço de Fechamento.

Valor de retorno

Se for bem sucedido, retorna o montante do lucro ou EMPTY_VALUE em caso de erro.

© 2000-2024, MetaQuotes Ltd.

MarginCheck

Obtém a quantidade de margem que é necessária para o funcionamento da negociação.

```
double  MarginCheck(
   const string        symbol,              // symbol
   ENUM_ORDER_TYPE     trade_operation,     // operation
   double              volume,              // volume
   double              price                // price
   ) const
```

Parâmetros

symbol

[in] Símbolo para a operação de negociação.

trade_operation

[in] Tipo de operação de negociação (enumerador ENUM_ORDER_TYPE).

volume

[in] Volume de operação de negociação.

price

[in] Preço de operação de negociação.

Valor de retorno

Quantidade de margem que é necessária para o funcionamento da negociação.

© 2000-2024, MetaQuotes Ltd.

FreeMarginCheck

Obtém a quantidade de margem livre remanescente após a operação de negociação.

```
double  FreeMarginCheck(
   const string        symbol,              // symbol
   ENUM_ORDER_TYPE     trade_operation,     // operation
   double              volume,              // volume
   double              price                // price
   ) const
```

Parâmetros

symbol

[in] Símbolo para a operação de negociação.

trade_operation

[in] Tipo de operação de negociação (enumerador ENUM_ORDER_TYPE).

volume

[in] Volume de operação de negociação.

price

[in] Preço de operação de negociação.

Valor de retorno

Quantidade de margem livre remanescente após a operação de negociação.

© 2000-2024, MetaQuotes Ltd.

MaxLotCheck

Obtém o volume máximo possível da operação de negociação.

```
double  MaxLotCheck(
   const string        symbol,              // symbol
   ENUM_ORDER_TYPE     trade_operation,     // operation
   double              price,               // price
   double              percent=100          // percent of available margin (0-100%), u
   ) const
```

Parâmetros

symbol

[in] Símbolo para a operação de negociação.

trade_operation

[in] Tipo de operação de negociação (enumeradorENUM_ORDER_TYPE).

price

[in] Preço de operação de negociação.

percent=100

[in] Porcentagem da margem disponível (0-100%), usado para operação de negociação.

Valor de retorno

O volume máximo possível da operação de negociação.

© 2000-2024, MetaQuotes Ltd.

CSymbolInfo

CSymbolInfo é uma classe para facilitar o acesso às propriedades do símbolo (ativo).

Descrição

A classe CSymbolInfo fornece acesso às propriedades do símbolo.

Declaração

```
class CSymbolInfo : public CObject
```

Título

```
#include <Trade\SymbolInfo.mqh>
```

Hierarquia de herança

CObject
  CSymbolInfo

Métodos de classe pelos grupos

Controlador	
Refresh	Atualiza os dados do símbolo
RefreshRates	Atualiza as cotações do símbolo
Propriedades	
Name	Obtém/define o nome do símbolo
Select	Obtém/define o flag do símbolo da "Observação do Mercado" (Market Watch)
IsSynchronized	Verifica a sincronização do símbolo com o servidor
Volumes	
Volume	Obtém o volume da última oferta
VolumeHigh	Obtém o volume máximo para um dia
VolumeLow	Obtém o volume mínimo para um dia
Miscelâneas	
Time	Obtém o tempo da última cotação
Spread	Obtém o valor do spread (em pontos)
SpreadFloat	Obtém o flag do spread flutuante
TicksBookDepth	Obtém a profundidade de ticks salvos (Livro de Ofertas)
Níveis	

© 2000-2024, MetaQuotes Ltd.

Controlador	
StopsLevel	Obtém o nível de stop mínimo para as ordens (em pontos)
FreezeLevel	Obtém a distância das operações de negociação congeladas (em pontos)
Preços Bid	
Bid	Obtém a atualização do preço Bid (preço de compra de quem entra vendido numa posição)
BidHigh	Obtém a máxima do preço Bid para um dia
BidLow	Obtém a mínima do preço Bid para um dia
Preços Ask	
Ask	Obtém a atualização do preço Ask (preço de venda de quem entra comprado numa posição)
AskHigh	Obtém a máxima do preço Ask para um dia
AskLow	Obtém a mínima do preço Ask para um dia
Preços	
Last	Obtém a atualização do preço Last (o último preço)
LastHigh	Obtém a máxima do preço Last para um dia
LastLow	Obtém a mínima do preço Last para um dia
Modos de negociação	
TradeCalcMode	Obtém o modo de cálculo dos custos do contrato
TradeCalcModeDescription	Obtém o modo de cálculo dos custos do contrato como uma string
TradeMode	Obtém o tipo de execução das ordens
TradeModeDescription	Obtém o tipo de execução de ordens como uma string
TradeExecution	Obtém o fechamento de modo de ofertas
TradeExecutionDescription	Obtém o fechamento de modo de ofertas como uma string
Swaps	
SwapMode	Obtém o modelo de cálculo Swap
SwapModeDescription	Obtém o modelo de cálculo Swap como uma string
SwapRollover3days	Obtém o dia do custo swap triplo
SwapRollover3daysDescription	Obtém o dia do custo swap triplo como uma string
Margens e flags	
MarginInitial	Obtém o valor da margem inicial

© 2000-2024, MetaQuotes Ltd.

Controlador	
MarginMaintenance	Obtém o valor da margem de manutenção
MarginLong	Obtém a taxa de margem necessária para posições compradas
MarginShort	Obtém a taxa de margem necessária para posições vendidas
MarginLimit	Obtém a taxa de margem necessária para ordens do tipo Limit
MarginStop	Obtém a taxa de margem necessária para ordens do tipo Stop
MarginStopLimit	Obtém a taxa de margem necessária para ordens do tipo StopLimit
TradeTimeFlags	Obtém os flags dos modos de validade das ordens permitidas
TradeFillFlags	Obtém os flags dos modos de preenchimento das ordens permitidas
Quantização	
Digits	Obtém o número de dígitos após o período
Point	Obtém o valor de um ponto
TickValue	Obtém o custo do tick (alteração mínima de preço)
TickValueProfit	Obtém o preço do tick calculado para uma posição lucrativa
TickValueLoss	Obtém o preço do tick calculado para uma posição perdedora
TickSize	Obtém a alteração mínima de preço
Tamanhos dos contratos	
ContractSize	Obtém a quantidade de contrato da negociação
LotsMin	Obtém o volume mínimo para fechar uma oferta
LotsMax	Obtém o volume máximo para fechar uma oferta
LotsStep	Obtém o passo mínimo de variação de volume para fechar uma oferta
LotsLimit	Obtém o volume máximo permitido de posições abertas e ordens pendentes (direção insensível) para um símbolo
Tamanhos de Swaps	
SwapLong	Obtém o valor swap da posição comprada
SwapShort	Obtém o valor swap da posição vendida
Propriedades de texto	
CurrencyBase	Obtém o nome da moeda do símbolo base
CurrencyProfit	Obtém o nome da moeda de lucro
CurrencyMargin	Obtém o nome da moeda de margem

© 2000-2024, MetaQuotes Ltd.

Controlador	
Bank	Obtém o nome da fonte da cotação atual
Descrição	Obtém a descrição da string do símbolo
Path	Obtém o caminho do diretório dos símbolos
Propriedades do símbolo	
SessionDeals	Obtém o número de ofertas na sessão atual
SessionBuyOrders	Obtém o número de ordens de Compra no momento
SessionSellOrders	Obtém o número de ordens de Venda no momento
SessionTurnover	Obtém o resumo do volume de negócios da sessão atual
SessionInterest	Obtém o resumo dos contratos em aberto da sessão atual
SessionBuyOrdersVolume	Obtém o atual volume de ordens de compra
SessionSellOrdersVolume	Obtém o atual volume das ordens de Venda
SessionOpen	Obtém o preço de abertura da sessão atual
SessionClose	Obtém o preço de fechamento da sessão atual
SessionAW	Obtém a média ponderada do preço da sessão atual
SessionPriceSettlement	Obtém o preço da liquidação da sessão atual
SessionPriceLimitMin	Obtém o preço mínimo da sessão atual
SessionPriceLimitMax	Obtém o preço máximo da sessão atual
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém o valor específico da propriedade tipo string
Funções de serviço	
NormalizePrice	Retorna o valor do preço, normalizado e utilizando as propriedades do símbolo

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza os dados do símbolo.

```
void  Refresh()
```

Valor de retorno

Nenhum.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

RefreshRates

Atualiza os dados das cotações dos símbolos.

```
bool  RefreshRates()
```

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível atualizar cotações.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Name

Obtém nome do símbolo (ativo).

```
string  Name() const
```

Valor de retorno

Nome do símbolo.

Name

Define o nome do símbolo.

```
bool  Name(string name)
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Select

Obtém o flag do símbolo da "Observação do Mercado" (Market Watch).

```
bool  Select() const
```

Valor de retorno

Obtém o flag do símbolo da "Observação do Mercado" (Market Watch)

Select

Define o flag do símbolo da "Observação do Mercado" (Market Watch).

```
bool  Select()
```

Valor de retorno

verdadeiro - em caso de sucesso, falso - se for incapaz de mudar o flag.

© 2000-2024, MetaQuotes Ltd.

IsSynchronized

Verifica a sincronização do símbolo com o servidor .

```
bool  IsSynchronized() const
```

Valor de retorno

verdadeiro - se o símbolo é sincronizado com o servidor, falso - se não.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Volume

Obtém o último volume de negócio.

```
long  Volume() const
```

Valor de retorno

Volume de último negócio.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

VolumeHigh

Obtém o volume máximo do dia.

```
long  VolumeHigh() const
```

Valor de retorno

O volume máximo do dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

VolumeLow

Obtém o volume mínimo do dia.

```
long  VolumeLow() const
```

Valor de retorno

Volume mínimo do dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Time

Obtém o tempo da última cotação.

```
datetime  Time() const
```

Valor de retorno

Tempo da última cotação.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Spread

Obtém o valor de spread (em pontos).

```
int  Spread() const
```

Valor de retorno

Obtém o valor de spread (em pontos).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SpreadFloat

Obtém o flag do spread flutuante.

```
bool  SpreadFloat() const
```

Valor de retorno

Flag do spread flutuante.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TicksBookDepth

Obtém a profundidade dos ticks salvos (Livro de Ofertas).

```
int  TicksBookDepth() const
```

Valor de retorno

Profundidade dos ticks salvos.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

StopsLevel

Obtém o nível de stop mínimo para as ordens (em pontos).

```
int  StopsLevel() const
```

Valor de retorno

O nível de stop mínimo para as ordens (em pontos).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

FreezeLevel

Obtém o nível de congelamento (em pontos).

```
int  FreezeLevel() const
```

Valor de retorno

Distância do nível de congelamento (em pontos).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Bid

Obtém a atualização do preço Bid (preço de compra para quem entra vendido numa posição)

```
double  Bid() const
```

Valor de retorno

Preço de compra Bid.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

BidHigh

Obtém a máxima do preço Bid para um dia.

```
double  BidHigh() const
```

Valor de retorno

A máxima do preço Bid para um dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

BidLow

Obtém a mínima do preço Bid para um dia.

```
double  BidLow() const
```

Valor de retorno

A mínima do preço Bid para um dia

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Ask

Obtém a atualização do preço Ask (preço de venda para quem entra comprado numa posição).

```
double  Ask() const
```

Valor de retorno

Atualização do preço Ask.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

AskHigh

Obtém a máxima do preço Ask para um dia.

```
double  AskHigh() const
```

Valor de retorno

Obtém a máxima do preço Ask para um dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

AskLow

Obtém a mínima do preço Ask para um dia.

```
double  AskLow() const
```

Valor de retorno

Mínima do preço Ask para um dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Last

Obtém a atualização do preço Last (o último preço).

```
double  Last() const
```

Valor de retorno

Atualização do preço Last.

© 2000-2024, MetaQuotes Ltd.

LastHigh

Obtém a máxima do preço Last para um dia.

```
double  LastHigh() const
```

Valor de retorno

A máxima do preço Last para um dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

LastLow

Obtém a mínima do preço Last para um dia

```
double  LastLow() const
```

Valor de retorno

A mínima do preço Last para um dia.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeCalcMode

Obtém o modo de cálculo dos custos do contrato.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Valor de retorno

Modo de cálculo do custo do contrato (valor do enumerador ENUM_SYMBOL_CALC_MODE).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeCalcModeDescription

Obtém o modo de cálculo dos custos do contrato como uma string.

```
string  TradeCalcModeDescription() const
```

Valor de retorno

Modo de cálculo do custo do contrato como uma string.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeMode

Obtém o tipo de execução de ordens.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Valor de retorno

Tipo de execução de ordens (valor do enumerador ENUM_SYMBOL_TRADE_MODE).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeModeDescription

Obtém o modo de negociação como uma string.

```
string  TradeModeDescription() const
```

Valor de retorno

Modo de negociação como uma string.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeExecution

Obtém o modo de execução da negociação.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Valor de retorno

Modo de execução da negociação (valor do enumerador ENUM_SYMBOL_TRADE_EXECUTION).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeExecutionDescription

Obtém a descrição do modo de execução da negociação como uma string.

```
string  TradeExecutionDescription() const
```

Valor de retorno

Modo de execução da negociação como uma string.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapMode

Obtém o modo de cálculo Swap.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Valor de retorno

Modo de cálculo swap (valor do enumerador ENUM_SYMBOL_SWAP_MODE).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapModeDescription

Obtém a descrição do modo swap como uma string.

```
string  SwapModeDescription() const
```

Valor de retorno

Descrição do modo swap como uma string.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapRollover3days

Obtém o dia do rolamento swap.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Valor de retorno

Dia do rolamento swap (valor do enumerador ENUM_DAY_OF_WEEK).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapRollover3daysDescription

Obtém o dia do rolamente swap como uma string.

```
string  SwapRollover3daysDescription() const
```

Valor de retorno

Dia do rolamento swap como uma string.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginInitial

Obtém o valor da margem inicial.

```
double  MarginInitial()
```

Valor de retorno

Valor da margem inicial.

Observação

Ele aponta a quantidade de margem (em moeda de margem do instrumento) que é analisada de um lote. Usado para verificar patrimônio do cliente, quando ele entra no mercado.

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginMaintenance

Obtém o valor da margem de manutenção.

```
double  MarginMaintenance()
```

Valor de retorno

Valor da margem de manutenção.

Observação

Ele aponta a quantidade de margem (em moeda de margem do instrumento) que é analisada de um lote. Usado para verificar o patrimônio líquido do cliente, quando o estado da conta for alterada. Se a margem de manutenção é igual a 0, então a margem inicial é usada.

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginLong

Obtém a taxa de margem necessária para posições compradas.

```
double  MarginLong() const
```

Valor de retorno

Taxa de margem necessária para posições compradas.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginShort

Obtém a taxa de margem necessária para posições vendidas.

```
double  MarginShort() const
```

Valor de retorno

Taxa de margem de necessária para posições vendidas.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginLimit

Obtém a taxa de margem necessária para ordens do tipo Limit.

```
double  MarginLimit() const
```

Valor de retorno

Taxa de margem necessária para ordens do tipo Limit.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginStop

Obtém a taxa de margem necessária para ordens do tipo Stop.

```
double  MarginStop() const
```

Valor de retorno

Taxa de margem necessária para ordens do tipo Stop.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

MarginStopLimit

Obtém a taxa de margem necessária para ordens do tipo StopLimit.

```
double  MarginStopLimit() const
```

Valor de retorno

Taxa de margem necessária para ordens do tipo StopLimit.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeTimeFlags

Obtém os flags dos modos de validade das ordens permitidas.

```
int  TradeTimeFlags() const
```

Valor de retorno

Flags dos modos de validade das ordens permitidas.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TradeFillFlags

Obtém os flags dos modos de preenchimento das ordens permitidas.

```
int  TradeFillFlags() const
```

Valor de retorno

Flags dos modos de preenchimento das ordens permitidas.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Digits

Obtém o número de dígitos após período.

```
int  Digits() const
```

Valor de retorno

Obtém o número de dígitos após período.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Point

Obtém o valor de um ponto.

```
double  Point() const
```

Valor de retorno

Valor de um ponto.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TickValue

Obtém o custo do tick (alteração mínima de preço).

```
double  TickValue() const
```

Valor de retorno

Custo do tick (alteração mínima de preço).

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TickValueProfit

Obtém o preço do tick calculado para uma posição lucrativa.

```
double  TickValueProfit() const
```

Valor de retorno

O preço do tick calculado para uma posição lucrativa.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TickValueLoss

Obtém o preço do tick calculado para uma posição perdedora.

```
double  TickValueLoss() const
```

Valor de retorno

Preço do tick calculado para uma posição perdedora.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

TickSize

Obtém a alteração mínima de preço.

```
double  TickSize() const
```

Valor de retorno

Alteração mínima de preço.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

ContractSize

Obtém a quantidade de contrato de negociação.

```
double  ContractSize() const
```

Valor de retorno

Quantidade de contrato de negociação.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

LotsMin

Obtém o volume mínimo para fechar uma oferta.

```
double  LotsMin() const
```

Valor de retorno

Volume mínimo para fechar uma oferta.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

LotsMax

Obtém o volume máximo para fechar uma oferta.

```
double  LotsMax() const
```

Valor de retorno

Volume máximo para fechar uma oferta.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

LotsStep

Obtém o passo mínimo de variação de volume para fechar uma oferta.

```
double  LotsStep() const
```

Valor de retorno

O passo mínimo de variação de volume para fechar uma oferta.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

LotsLimit

Obtém o volume máximo permitido de posições abertas e ordens pendentes (direção insensível) para um símbolo.

```
double  LotsLimit() const
```

Valor de retorno

O volume máximo permitido de posições abertas e ordens pendentes (direção insensível) para um símbolo.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapLong

Obtém o valor swap de uma posição comprada.

```
double  SwapLong() const
```

Valor de retorno

Valor swap de uma posição comprada.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SwapShort

Obtém o valor swap da posição vendida.

```
double  SwapShort() const
```

Valor de retorno

O valor swap da posição vendida.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyBase

Obtém o nome do símbolo da moeda base.

```
string  CurrencyBase() const
```

Valor de retorno

Nome do símbolo da moeda base.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyProfit

Obtém o nome da moeda de lucro.

```
string  CurrencyProfit() const
```

Valor de retorno

Nome da moeda de lucro.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

CurrencyMargin

Obtém o nome da moeda de margem.

```
string  CurrencyMargin() const
```

Valor de retorno

Nome da moeda de margem.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Bank

Obtém o nome da fonte da cotação atual.

```
string  Bank() const
```

Valor de retorno

Nome da fonte da cotação atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Descrição

Obtém a descrição da string do símbolo.

```
string  Description() const
```

Valor de retorno

Descrição da string do símbolo.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

Path

Obtém o caminho do diretório dos símbolos.

```
string  Path() const
```

Valor de retorno

Obtém o caminho do diretório dos símbolos.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionDeals

Obtém o número de ofertas na sessão atual.

```
long  SessionDeals() const
```

Valor de retorno

Número de ofertas na sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionBuyOrders

Obtém o número de ordens de Compra no momento.

```
long  SessionBuyOrders() const
```

Valor de retorno

Número de ordens de compra no momento.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionSellOrders

Obtém o número de ordens de Venda no momento.

```
long  SessionSellOrders() const
```

Valor de retorno

Número de ordens de Venda no momento.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionTurnover

Obtém resumo do volume de negócios da sessão atual

```
double  SessionTurnover() const
```

Valor de retorno

Resumo do volume de negócios da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionInterest

Obtém o resumo dos contratos em aberto da sessão atual.

```
double  SessionInterest() const
```

Valor de retorno

O resumo dos contratos em aberto da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionBuyOrdersVolume

Obtém o atual volume de ordens de compra.

```
double  SessionBuyOrdersVolume() const
```

Valor de retorno

Volume atual de ordens de compra.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionSellOrdersVolume

Obtém o atual volume das ordens de venda.

```
double  SessionSellOrdersVolume() const
```

Valor de retorno

Volume atual das ordens de venda.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionOpen

Obtém o preço de abertura da sessão atual.

```
double  SessionOpen() const
```

Valor de retorno

Preço de abertura da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionClose

Obtém o preço de fechamento da sessão atual.

```
double  SessionClose() const
```

Valor de retorno

Preço de fechamento da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionAW

Obtém a média ponderada do preço da sessão atual

```
double  SessionAW() const
```

Valor de retorno

A média ponderada do preço da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceSettlement

Obtém o preço de liquidação da sessão atual.

```
double  SessionPriceSettlement() const
```

Valor de retorno

O preço de liquidação da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceLimitMin

Obtém o preço mínimo da sessão atual.

```
double  SessionPriceLimitMin() const
```

Valor de retorno

Preço mínimo da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

SessionPriceLimitMax

Obtém o preço máximo da sessão atual.

```
double  SessionPriceLimitMax() const
```

Valor de retorno

Preço máximo da sessão atual.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // property ID
   long&                     var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade tipo integer (valor do enumerador ENUM_SYMBOL_INFO_INTEGER).

var

[out] Referente a variável do tipo integer para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // property ID
   double&                  var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade de tipo double (valor do enumerador ENUM_SYMBOL_INFO_DOUBLE).

var

[out] Referente a variável do tipo double para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // property ID
   string&                  var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade do texto.

var

[out] Referente a variável do tipo string para colocar resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

NormalizePrice

Retorna o valor do preço, normalizado e utilizando as propriedades do símbolo.

```
double  NormalizePrice(
   double  price   // price
   ) const
```

Parâmetros

price

[in] Preço.

Valor de retorno

Preço normalizado.

Observação

O símbolo deve ser selecionado pelo método Name.

© 2000-2024, MetaQuotes Ltd.

COrderInfo

COrderInfo é uma classe para facilitar o acesso às propriedades das ordens pendentes.

Descrição

A classe COrderInfo fornece acesso às propriedades das ordens pendente.

Declaração

```
class COrderInfo : public CObject
```

Título

```
#include <Trade\OrderInfo.mqh>
```

Hierarquia de herança

CObject
  COrderInfo

Métodos de classe pelos grupos

O acesso a propriedades do tipo integer	
Ticket	Obtém o ticket de uma ordem, previamente selecionada para acessar
TimeSetup	Obtém o tempo de colocação das ordens
TimeSetupMsc	Recebe o tempo de colocação de uma ordem em milissegundos desde 01.01.1970
OrderType	Obtém o tipo de ordem
OrderTypeDescription	Obtém o tipo de ordem como uma string
State	Obtém o estado da ordem
StateDescription	Obtém o estado da ordem como uma string
TimeExpiration	Obtém o tempo de expiração da ordem
TimeDone	Obtém o tempo de execução ou cancelamento da ordem
TimeDoneMsc	Recebe a execução da ordem ou o tempo de cancelamento em milissegundos desde 01.01.1970
TypeFilling	Obtém o tipo de execução da ordem pela remanescente
TypeFillingDescription	Obtém o tipo de execução da ordem pela remanescente como uma string
TypeTime	Obtém o tipo de ordem pelo tempo de expiração

© 2000-2024, MetaQuotes Ltd.

O acesso a propriedades do tipo integer	
TypeTimeDescription	Obtém o tipo de ordem pelo tempo de expiração como uma string
Magic	Obtém a identificação do expert que coloca a ordem
PositionId	Obtém a identificação da posição
O acesso a propriedades do tipo double	
VolumeInitial	Obtém o volume inicial da ordem
VolumeCurrent	Obtém o volume não preenchido da ordem
PriceOpen	Obtém o preço da ordem
StopLoss	Obtém o Stop Loss da ordem
TakeProfit	Obtém o Take Profit da ordem
PriceCurrent	Obtém o preço atual do símbolo (ativo) da ordem
PriceStopLimit	Obtém o preço configurado da ordem tipo "limit"
O acesso a propriedades de texto	
Symbol	Obtém o nome do símbolo (ativo) da ordem
Comment	Obtém o comentário da ordem
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém valor específico da propriedade tipo string
State	
StoreState	Salva os parâmetros da ordem
CheckState	Verifica os parâmetros atuais comparando com os parâmetros salvos
Seleção	
Select	Seleciona uma ordem pelo ticket para mais acesso às propriedades
SelectByIndex	Seleciona uma ordem de índice para acessar as propriedades

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Ticket

Obtém o ticket de uma ordem, previamente selecionado para acesso usando o método Select.

```
ulong  Ticket() const
```

Valor de retorno

Se bem sucedido, retorna o ticket da ordem, de outra forma retorna ULONG_MAX.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeSetup

Obtém o tempo de colocação da ordem.

```
datetime  TimeSetup() const
```

Valor de retorno

Tempo de colocação da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeSetupMsc

Recebe o tempo para execução de uma ordem em milissegundos desde 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Valor de retorno

O tempo para execução de uma ordem em milissegundos desde 01.01.1970.

Observação

A ordem deve ser preliminarmente selecionada para acesso usando o método Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

OrderType

Obtém o tipo de ordem.

```
ENUM_ORDER_TYPE  OrderType()
```

Valor de retorno

Tipo de ordem (valor do enumerador ENUM_ORDER_TYPE).

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Obtém o tipo de ordem como uma string.

```
string  TypeDescription() const
```

Valor de retorno

Tipo de ordem como uma string.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

State

Obtém o estado da ordem.

```
ENUM_ORDER_STATE  State() const
```

Valor de retorno

Estado da ordem (valor do enumerador ENUM_ORDER_STATE).

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StateDescription

Obtém o estado da ordem como uma string.

```
string  StateDescription() const
```

Valor de retorno

Estado da ordem como uma string.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeExpiration

Obtém o tempo de expiração da ordem.

```
datetime  TimeExpiration() const
```

Valor de retorno

Tempo de expiração da ordem, configurado na colocação da mesma.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeDone

Obtém o tempo de execução ou cancelamento da ordem.

```
datetime  TimeDone() const
```

Valor de retorno

Tempo de execução ou cancelamento da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeDoneMsc

Recebe tempo de execução ou cancelamento da ordem em milissegundos desde 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Valor de retorno

Tempo de execução ou cancelamento da ordem em milissegundos desde 01.01.1970.

Observação

A ordem deve ser preliminarmente selecionada para acesso usando o método Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeFilling

Obtém o tipo de preenchimento da ordem.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valor de retorno

Tipo de preenchimento da ordem (valor do enumerador ENUM_ORDER_TYPE_FILLING).

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeFillingDescription

Obtém o tipo de preenchimento da ordem como uma string.

```
string  TypeFillingDescription() const
```

Valor de retorno

Tipo de preenchimento da ordem como uma string.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeTime

Obtém o tipo de ordem pelo tempo de expiração.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valor de retorno

Tipo de ordem pelo tempo da expiração.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeTimeDescription

Obtém o tipo de ordem pelo tempo de expiração como uma string.

```
string  TypeTimeDescription() const
```

Valor de retorno

Tipo de ordem pelo tempo de expiração como uma string.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Magic

Obtém a identificação do expert que coloca a ordem.

```
long  Magic() const
```

Valor de retorno

ID do expert que coloca a ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PositionId

Obtém a identificação da posição.

```
long  PositionId() const
```

Valor de retorno

ID da posição em que a ordem foi envolvida.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

VolumeInitial

Obtém o volume inicial da ordem.

```
double  VolumeInitial() const
```

Valor de retorno

Volume inicial da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

VolumeCurrent

Obtém o volume não preenchido da ordem.

```
double  VolumeCurrent() const
```

Valor de retorno

Volume não preenchido da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Obtém o preço da ordem.

```
double  PriceOpen() const
```

Valor de retorno

Preço para colocar a ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Obtém Stop Loss da ordem.

```
double  StopLoss() const
```

Valor de retorno

Stop Loss da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Obtém o Take Profit da ordem.

```
double  TakeProfit() const
```

Valor de retorno

Take Profit da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Obtém o preço atual pelo símbolo (ativo) da ordem.

```
double  PriceCurrent() const
```

Valor de retorno

Preço atual pelo símbolo (ativo) da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceStopLimit

Obtém o preço configurado da ordem tipo "limit".

```
double  PriceStopLimit() const
```

Valor de retorno

Preço configurado da ordem tipo "limit".

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém o nome do símbolo (ativo) da ordem.

```
string  Symbol() const
```

Valor de retorno

Nome do símbolo (ativo) da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Comment

Obtém o comentário da ordem.

```
string  Comment() const
```

Valor de retorno

Comentário da ordem.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // property ID
   long&                        var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade tipo integer (valor do enumerador ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Referente a variável do tipo integer para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // property ID
   double&                     var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade de tipo double (valor do enumerador ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Referente a variável do tipo double para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // property ID
   string&                     var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade do texto.

var

[out] Referente a variável do tipo string para colocar resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StoreState

Salva os parâmetros da ordem.

```
void  StoreState()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CheckState

Verifica os parâmetros atuais comparando com os parâmetros salvos.

```
bool  CheckState()
```

Valor de retorno

verdadeiro - se os parâmetros de ordem mudaram desde a última chamada pelo método StoreState(), caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Select

Seleciona uma ordem pelo ticket para mais acessos às propriedades.

```
bool  Select(
   ulong  ticket   // order ticket
   )
```

Valor de retorno

verdadeiro - em caso de sucesso, Falso - se não for possível selecionar a ordem.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Seleciona a ordem pelo índice para mais acesso às propriedades.

```
bool  SelectByIndex(
   int  index   // order index
   )
```

Valor de retorno

verdadeiro - em caso de sucesso, Falso - se não for possível selecionar a ordem.

© 2000-2024, MetaQuotes Ltd.

CHistoryOrderInfo

CHistoryOrderInfo é uma classe para facilitar o acesso às propriedades da ordem no histórico.

Descrição

A classe CHistoryOrderInfo possibilita o fácil acesso às propriedades da ordem no histórico.

Declaração

```
class CHistoryOrderInfo : public CObject
```

Título

```
#include <Trade\HistoryOrderInfo.mqh>
```

Hierarquia de herança

CObject
  CHistoryOrderInfo

Métodos de classe pelos grupos

O acesso a propriedades do tipo integer	
TimeSetup	Obtém o tempo de colocação das ordens
TimeSetupMsc	Recebe o tempo de colocar uma ordem em milissegundos desde 01.01.1970
OrderType	Obtém o tipo de ordem
OrderTypeDescription	Obtém o tipo de ordem como uma string
State	Obtém o estado da ordem
StateDescription	Obtém o estado da ordem como uma string
TimeExpiration	Obtém o tempo de expiração da ordem
TimeDone	Obtém o tempo de execução ou cancelamento da ordem
TimeDoneMsc	Recebe a execução da ordem ou o tempo de cancelamento em milissegundos desde 01.01.1970
TypeFilling	Obtém o tipo de execução da ordem pela remanescente
TypeFillingDescription	Obtém o tipo de execução da ordem pela remanescente como uma string
TypeTime	Obtém o tipo de ordem pelo tempo de expiração
TypeTimeDescription	Obtém o tipo de ordem pelo tempo de expiração como uma string
Magic	Obtém a identificação do expert que coloca a ordem

© 2000-2024, MetaQuotes Ltd.

O acesso a propriedades do tipo integer	
PositionId	Obtém a identificação da posição
O acesso a propriedades do tipo double	
VolumeInitial	Obtém o volume inicial da ordem
VolumeCurrent	Obtém o volume não preenchido da ordem
PriceOpen	Obtém o preço da ordem
StopLoss	Obtém o Stop Loss da ordem
TakeProfit	Obtém o Take Profit da ordem
PriceCurrent	Obtém o preço atual do símbolo (ativo) da ordem
PriceStopLimit	Obtém o preço configurado da ordem tipo "limit"
O acesso a propriedades de texto	
Symbol	Obtém o símbolo (ativo) da ordem
Comment	Obtém o comentário da ordem
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém valor específico da propriedade tipo string
Seleção	
Ticket	Obtém o ticket/seleciona a ordem
SelectByIndex	Seleciona a ordem pelo índice

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

TimeSetup

Obtém o tempo de colocação da ordem.

```
datetime  TimeSetup() const
```

Valor de retorno

Tempo de colocação da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeSetupMsc

Recebe o tempo de colocar uma ordem para execução em milissegundos desde 01.01.1970.

```
ulong  TimeSetupMsc() const
```

Valor de retorno

O tempo de colocar uma ordem para execução em milissegundos desde 01.01.1970.

Observação

Ordem histórica deve ser preliminarmente selecionada para acesso usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

OrderType

Obtém o tipo de ordem.

```
ENUM_ORDER_TYPE  OrderType() const
```

Valor de retorno

Tipo de ordem (valor do enumerador ENUM_ORDER_TYPE).

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Obtém o tipo de ordem como uma string.

```
string  TypeDescription() const
```

Valor de retorno

Tipo de ordem como uma string.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

State

Obtém o estado da ordem.

```
ENUM_ORDER_STATE  State() const
```

Valor de retorno

Estado da ordem (valor do enumerador ENUM_ORDER_STATE).

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StateDescription

Obtém o estado da ordem como uma string.

```
string  StateDescription() const
```

Valor de retorno

Estado da ordem como uma string.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeExpiration

Obtém o tempo de expiração da ordem.

```
datetime  TimeExpiration() const
```

Valor de retorno

Tempo de expiração da ordem, definido quando da sua colocação.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeDone

Obtém o tempo de execução ou cancelamento da ordem.

```
datetime  TimeDone() const
```

Valor de retorno

Tempo de execução ou cancelamento da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeDoneMsc

Recebe de execução da ordem ou o tempo de cancelamento em milissegundos desde 01.01.1970.

```
ulong  TimeDoneMsc() const
```

Valor de retorno

Execução da ordem ou o tempo de cancelamento em milissegundos desde 01.01.1970.

Observação

Ordem histórica deve ser preliminarmente selecionada para acesso usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeFilling

Obtém o tipo de execução da ordem pela remanescente.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Valor de retorno

Tipo de execução da ordens pela remanescente (valor do enumerador ENUM_ORDER_TYPE_FILLING).

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeFillingDescription

Obtém o tipo de execução da ordem pela remanescente como uma string.

```
string  TypeFillingDescription() const
```

Valor de retorno

Tipo de execução da ordem pela remanescente como uma string.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeTime

Obtém o tipo de ordem pelo tempo de expiração.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Valor de retorno

Tipo de ordem pelo tempo de expiração (valor do enumerador ENUM_ORDER_TYPE_TIME).

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeTimeDescription

Obtém o tipo de ordem pelo tempo de expiração como uma string.

```
string  TypeTimeDescription() const
```

Valor de retorno

Tipo de ordem pelo tempo de expiração como uma string.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Magic

Obtém a identificação do expert que coloca a ordem.

```
long  Magic() const
```

Valor de retorno

ID do Expert Advisor que coloca a ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PositionId

Obtém a identificação da posição.

```
long  PositionId() const
```

Valor de retorno

ID da posição em que a ordem foi envolvida.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

VolumeInitial

Obtém o volume inicial da ordem.

```
double  VolumeInitial() const
```

Valor de retorno

Volume inicial da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

VolumeCurrent

Obtém o volume não preenchido da ordem.

```
double  VolumeCurrent() const
```

Valor de retorno

Volume não preenchido da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Obtém o preço da ordem.

```
double  PriceOpen() const
```

Valor de retorno

Preço para colocar a ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Obtém o preço do Stop Loss da ordem.

```
double  StopLoss() const
```

Valor de retorno

Preço do Stop Loss da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Obtém o preço do Take Profit da ordem.

```
double  TakeProfit() const
```

Valor de retorno

O preço do Take Profit da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Obtém o preço atual do símbolo (ativo) da ordem.

```
double  PriceCurrent() const
```

Valor de retorno

O preço atual do símbolo (ativo) da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceStopLimit

Obtém o preço da ordem tipo "limit".

```
double  PriceStopLimit() const
```

Valor de retorno

Preço da ordem tipo "limit".

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém o nome do símbolo (ativo) da ordem.

```
string  Symbol() const
```

Valor de retorno

Nome do símbolo (ativo) da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Comment

Obtém o comentário da ordem.

```
string  Comment() const
```

Valor de retorno

Comentário da ordem.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // property ID
   long&                        var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade tipo integer (valor do enumerador ENUM_ORDER_PROPERTY_INTEGER).

var

[out] Referente a variável do tipo integer para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // property ID
   double&                     var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade de tipo double (valor do enumerador ENUM_ORDER_PROPERTY_DOUBLE).

var

[out] Referente a variável do tipo double para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // property ID
   string&                     var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID de propriedade de texto (valor do enumerador ENUM_ORDER_PROPERTY_STRING).

var

[out] Referente a variável do tipo string para colocar resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A ordem histórica deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Ticket (Método Get)

Obtém o ticket da ordem.

```
ulong  Ticket() const
```

Valor de retorno

Ticket da ordem.

Ticket (Método Set)

Selecione a ordem para mais trabalhos.

```
void  Ticket(
   ulong  ticket      // order ticket
   )
```

Parâmetros

ticket

[in] Ticket da Ordem

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Seleciona a ordem pelo índice para mais acesso às propriedades.

```
bool  SelectByIndex(
   int  index   // order index
   )
```

Valor de retorno

verdadeiro - em caso de sucesso, Falso - se não for possível selecionar a ordem.

© 2000-2024, MetaQuotes Ltd.

CPositionInfo

CPositionInfo é uma classe para facilitar o acesso às propriedades de abertura de posição.

Descrição

A clase CPositionInfo possibilita o fácil acesso às propriedades de abertura de posição.

Declaração

```
class CPositionInfo : public CObject
```

Título

```
#include <Trade\PositionInfo.mqh>
```

Hierarquia de herança

CObject
  CPositionInfo

Métodos de classe pelos grupos

O acesso a propriedades do tipo integer	
Time	Obtém o tempo de abertura da posição
TimeMsc	Recebe o tempo da abertura de posição em milissegundos desde 01.01.1970
TimeUpdate	Recebe o tempo de mudança de posição em segundos desde 01.01.1970
TimeUpdateMsc	Recebe o tempo de mudança de posição em milissegundos desde 01.01.1970
PositionType	Obtém o tipo de posição
TypeDescription	Obtém o tipo de posição como uma string
Magic	Obtém a identificação do expert que abre a posição
Identificador	Obtém a identificação da posição
O acesso a propriedades do tipo double	
Volume	Obtém o volume da posição
PriceOpen	Obtém o preço de abertura da posição
StopLoss	Obtém o preço do Stop Loss da posição
TakeProfit	Obtém o preço do Take Profit da posição

© 2000-2024, MetaQuotes Ltd.

O acesso a propriedades do tipo integer	
PriceCurrent	Obtém o preço atual da posição pelo símbolo (ativo)
Comissão	Obtém o valor da comissão pelo posição
Swap	Obtém o valor de swap pela posição
Profit	Obtém a quantidade do lucro atual pela posição
O acesso a propriedades de texto	
Symbol	Obtém o nome do símbolo (ativo) da posição
Comment	Obtém o comentário da posição
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém o valor específico da propriedade tipo string
Seleção	
Select	Seleciona a posição
SelectByIndex	Seleciona a posição pelo índice
SelectByMagic	Seleciona a posição segundo o nome do instrumento e o número mágico para trabalhos futuros.
SelectByTicket	Seleciona a posição segundo o bilhete.
State	
StoreState	Salva os parâmetros da posição
CheckState	Verifica os parâmetros atuais comparando com os parâmetros salvos

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Time

Obtém o tempo de abertura de posição.

```
datetime  Time() const
```

Valor de retorno

Tempo de abertura da posição.

Observação

A posição deve ser seleccionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeMsc

Recebe tempo de abertura posição em milissegundos desde 01.01.1970.

```
ulong  TimeMsc() const
```

Valor de retorno

Tempo de abertura posição em milissegundos desde 01.01.1970.

Observação

Posição deve ser preliminarmente selecionada para acesso usando os métodos Select (pelo símbolo) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeUpdate

Recebe o tempo da mudança de posição em segundos, desde 01.01.1970.

```
datetime  TimeUpdate() const
```

Valor de retorno

Tempo da mudança de posição em segundos, desde 01.01.1970.

Observação

Posição deve ser preliminarmente selecionada para acesso usando os métodos Select (pelo símbolo) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeUpdateMsc

Recebe o tempo da mudança de posição em milissegundos desde 01.01.1970.

```
ulong  TimeUpdateMsc() const
```

Valor de retorno

O tempo da mudança de posição em milissegundos desde 01.01.1970.

Observação

Posição deve ser preliminarmente selecionada para acesso usando os métodos Select (pelo símbolo) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PositionType

Obtém o tipo de posição.

```
ENUM_POSITION_TYPE  PositionType() const
```

Valor de retorno

Tipo de posição (valor do enumerador ENUM_POSITION_TYPE).

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Obtém o tipo de posição como uma string.

```
string  TypeDescription() const
```

Valor de retorno

Tipo de posição como uma string.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Magic

Obtém a ID do Expert Advisor para abertura da posição.

```
long  Magic() const
```

Valor de retorno

ID do Expert Advisor para abertura da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Identificador

Obtém a identificação da posição.

```
long  Identifier() const
```

Valor de retorno

ID da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Volume

Obtém o volume da posição.

```
double  Volume() const
```

Valor de retorno

Volume da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceOpen

Obtém o preço de abertura da posição.

```
double  PriceOpen() const
```

Valor de retorno

Preço de abertura da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

StopLoss

Obtém o preço do Stop Loss da posição.

```
double  StopLoss() const
```

Valor de retorno

O preço do Stop Loss da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TakeProfit

Obtém o preço do Take Profit da posição.

```
double  TakeProfit() const
```

Valor de retorno

O preço do Take Profit da posição

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PriceCurrent

Obtém o preço atual pela posição do símbolo (ativo).

```
double  PriceCurrent() const
```

Valor de retorno

Preço atual pela posição do símbolo (ativo).

© 2000-2024, MetaQuotes Ltd.

Comissão

Obtém o valor de comissão da posição.

```
double  Commission() const
```

Valor de retorno

Valor de comissão da posição (em moeda do depósito).

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Swap

Obtém o valor swap da posição.

```
double  Swap() const
```

Valor de retorno

Valor swap da posição (em moeda de depósito).

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Profit

Obtém a quantidade de lucro atual da posição.

```
double  Profit() const
```

Valor de retorno

Quantidade de lucro atual da posição (em moeda de depósito).

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém o nome do símbolo (ativo) da posição.

```
string  Symbol() const
```

Valor de retorno

Nome do símbolo (ativo) da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Comment

Obtém o comentário da posição.

```
string  Comment() const
```

Valor de retorno

Comentário da posição.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // property ID
   long&                           var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade tipo integer (valor do enumerador ENUM_POSITION_PROPERTY_INTEGER).

var

[out] Referente a variável do tipo integer para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // property ID
   double&                        var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade de tipo double (valor do enumerador ENUM_POSITION_PROPERTY_DOUBLE).

var

[in] Referente a variável de tipo double para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // property ID
   string&                        var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID de propriedade de texto (valor do enumerador ENUM_POSITION_PROPERTY_STRING).

var

[out] Referente a variável do tipo string para colocar resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A posição deve ser selecionada usando os métodos Select (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Select

Selecione a posição para mais trabalho.

```
bool  Select(
   const string  symbol       // symbol
   )
```

Parâmetros

symbol

[in] Símbolo (ativo) da posição selecionada.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Seleciona a posição pelo índice para mais acesso às propriedades.

```
bool  SelectByIndex(
   int  index   // position index
   );
```

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível selecionar a posição.

© 2000-2024, MetaQuotes Ltd.

Seleciona a posição segundo o nome do instrumento e o número mágico para trabalhos futuros.

```
bool  SelectByMagic(
   const string  symbol, // nome do instrumento
   const ulong   magic   // número mágico
   );
```

Parâmetros

symbol

[in] Nome do instrumento.

magic

[in] Número mágico da posição.

valor de retorno

Retorna true se for bem-sucedido ou false no caso de a posição não estar selecionada..

© 2000-2024, MetaQuotes Ltd.

SelectByTicket

Seleciona a posição segundo um bilhete para trabalhos futuros.

```
bool  SelectByTicket(
   ulong  ticket      // bilhete da posição
   )
```

Parâmetros

ticket

[in] Bilhete da posição.

valor de retorno

Retorna true se for bem-sucedido ou false no caso de a posição não estar selecionada..

© 2000-2024, MetaQuotes Ltd.

StoreState

Salva os parâmetros da posição.

```
void  StoreState()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CheckState

Verifica os parâmetros atuais comparando com os parâmetros salvos.

```
bool  CheckState()
```

Valor de retorno

verdadeiro - se os parâmetros de posição mudaram desde a última chamada do método StoreState(), caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

CDealInfo

CDealInfo é uma classe para facilitar o acesso às propriedades das ofertas.

Descrição

A classe CDealInfo possibilita o acesso às propriedades das ofertas.

Declaração

```
class CDealInfo : public CObject
```

Título

```
#include <Trade\DealInfo.mqh>
```

Hierarquia de herança

CObject
  CDealInfo

Métodos de classe pelos grupos

O acesso a propriedades do tipo integer	
Order	Obtém a ordem pela qual a oferta é executada
Time	Obtém o tempo de execução da oferta
TimeMsc	Recebe o tempo de uma oferta executada em milissegundos desde 01.01.1970
DealType	Obtém o tipo de oferta
TypeDescription	Obtém o tipo de oferta como uma string
Entry	Obtém a direção da oferta
EntryDescription	Obtém a direção da oferta como uma string
Magic	Obtém a ID do expert que executou a oferta
PositionId	Obtém a ID da posição em que a oferta foi envolvida
O acesso a propriedades do tipo double	
Volume	Obtém o volume da oferta
Price	Obtém o preço da oferta
Commision	Obtém o valor da comissão pela oferta
Swap	Obtém o montante swap quando posição é fechada
Profit	Obtém o resultado financeiro da oferta

© 2000-2024, MetaQuotes Ltd.

O acesso a propriedades do tipo integer	
O acesso a propriedades de texto	
Symbol	Obtém o nome do símbolo da oferta
Comment	Obtém o comentário da oferta
Acesso a funções API MQL5	
InfoInteger	Obtém o valor específico da propriedade tipo integer
InfoDouble	Obtém valor específico da propriedade tipo double
InfoString	Obtém valor específico da propriedade tipo string
Seleção	
Ticket	Obtém o ticket/seleciona a oferta
SelectByIndex	Seleciona a oferta pelo índice

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Order

Obtém a ordem pela qual a oferta é executada.

```
long  Order() const
```

Valor de retorno

Ordem pela qual a oferta é executada.

Observação

A oferta deve ser selecionado usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Time

Obtém o tempo de execução da oferta.

```
datetime  Time() const
```

Valor de retorno

Tempo de execução da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TimeMsc

Recebe o tempo de execução da oferta em milissegundos desde 01.01.1970.

```
ulong  TimeMsc() const
```

Valor de retorno

O tempo de execução da oferta em milissegundos desde 01.01.1970.

Observação

A oferta deve ser preliminarmente selecionado para acesso usando o método Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

DealType

Obtém o tipo de oferta.

```
ENUM_DEAL_TYPE  DealType() const
```

Valor de retorno

Tipo de oferta (valor do enumerador ENUM_DEAL_TYPE).

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

TypeDescription

Obtém o tipo de oferta como uma string.

```
string  TypeDescription() const
```

Valor de retorno

Tipo de oferta como uma string.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Entry

Obtém a direção da oferta.

```
ENUM_DEAL_ENTRY  Entry() const
```

Valor de retorno

Direção da oferta (valor do enumerador ENUM_DEAL_ENTRY.).

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

EntryDescription

Obtém a direção da oferta como uma string.

```
string  EntryDescription() const
```

Valor de retorno

Direção da oferta como uma string.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Magic

Obtém a ID do Expert Advisor que executou a oferta.

```
long  Magic() const
```

Valor de retorno

ID do Expert Advisor, que executou a oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

PositionId

Obtém a identificação da posição em que a oferta foi envolvida.

```
long  PositionId() const
```

Valor de retorno

ID da posição em que a oferta foi envolvida.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Volume

Obtém o volume da oferta.

```
double  Volume() const
```

Valor de retorno

Volume da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Price

Obtém o preço da oferta.

```
double  Price() const
```

Valor de retorno

Preço da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Comissão

Obtém o valor de comissão da oferta.

```
double  Commission() const
```

Valor de retorno

Valor de comissão da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Swap

Obtém a quantidade swap quando posição é fechada.

```
double  Swap() const
```

Valor de retorno

Quantidade swap quando a posição é fechada.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Profit

Obtém o resultado financeiro da oferta.

```
double  Profit() const
```

Valor de retorno

O resultado financeiro da oferta (em moeda de depósito).

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Symbol

Obtém o nome do símbolo da oferta.

```
string  Symbol() const
```

Valor de retorno

Nome do símbolo da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Comment

Obtém o comentário da oferta.

```
string  Comment() const
```

Valor de retorno

Comentário da oferta.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Obtém o valor específico da propriedade tipo integer.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // property ID
   long&                       var           // reference to variable
   ) const
```

Parâmetros

prop_id

[In] ID da propriedade tipo integer (valor do enumerador ENUM_DEAL_PROPERTY_INTEGER).

var

[out] Referente a variável do tipo integer para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoDouble

Obtém o valor específico da propriedade do tipo double.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // property ID
   double&                    var           // reference to variable
   ) const
```

Parâmetros

prop_id

[in] ID da propriedade do tipo double (valor do enumerador ENUM_DEAL_PROPERTY_DOUBLE).

var

[in] Referente a variável de tipo double para colocar o resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

InfoString

Obtém o valor específico da propriedade tipo string.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // property ID
   string&                    var           // reference to variable
   ) const
```

Parâmetros

prop_id

[In] ID da propriedade de texto (valor do enumerador ENUM_DEAL_PROPERTY_STRING).

var

[out] Referente a variável do tipo string para colocar resultado.

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não for possível obter o valor da propriedade.

Observação

A oferta deve ser selecionada usando os métodos Ticket (pelo ticket) ou SelectByIndex (pelo índice).

© 2000-2024, MetaQuotes Ltd.

Ticket (Método Get)

Obtém o ticket da oferta.

```
ulong  Ticket() const
```

Valor de retorno

Ticket da oferta.

Ticket (Método Set)

Selecione a posição para mais trabalho.

```
void  Ticket(
   ulong  ticket      // ticket
   )
```

Parâmetros

ticket

[in] Ticket da oferta.

© 2000-2024, MetaQuotes Ltd.

SelectByIndex

Seleciona a oferta pelo índice para mais acessos às propriedades.

```
bool  SelectByIndex(
   int  index   // order index
   )
```

Valor de retorno

verdadeiro - em caso de sucesso, falso - se não foi possível selecionar a oferta.

© 2000-2024, MetaQuotes Ltd.

CTrade

CTrade é uma classe para facilitar o acesso às funções de negociação.

Descrição

A classe CTrade oferece facilidade de acesso às funções de negociação.

Declaração

```
class CTrade : public CObject
```

Título

```
#include <Trade\Trade.mqh>
```

Hierarquia de herança

CObject
  CTrade

Descendentes diretos

CExpertTrade

Métodos de classe pelos grupos

Definição dos parâmetros	
LogLevel	Define nível de entrada
SetExpertMagicNumber	Define a ID do expert
SetDeviationInPoints	Define o desvio permitido
SetTypeFilling	Define o preenchimento do tipo de ordem
SetTypeFillingBySymbol	Define o tipo de ordem segundo a execução de acordo com as configurações do símbolo especificado
SetAsyncMode	Define o modo assíncrono para operações de negociação
SetMarginMode	Define o modo de cálculo da margem de acordo com as configurações da conta atual
As operações com ordens	
OrderOpen	Coloca a ordem pendente com parâmetros definidos
OrderModify	Modifica os parâmetros da ordem pendente
OrderDelete	Exclui a ordem pendente
As operações com posições	
PositionOpen	Abre a posição com parâmetros definidos

© 2000-2024, MetaQuotes Ltd.

Definição dos parâmetros	
PositionModify	Altera os parâmetros da posição segundo quer o símbolo, quer o bilhete da posição
PositionClose	Fecha a posição segundo o símbolo especificado
PositionClosePartial	Fecha parte da posição cujo símbolo ou bilhete é especificado
PositionCloseBy	Fecha a posição com o bilhete indicado da posição com direção oposta.
Métodos adicionais	
Buy	Abre uma posição comprada com parâmetros determinados
Sell	Abre uma posição vendida com parâmetros determinados
BuyLimit	Coloca a ordem pendente do tipo Buy Limit com parâmetros determinados
BuyStop	Coloca a ordem pendente do tipo Buy Stop com parâmetros determinados
SellLimit	Coloca a ordem pendente do tipo Sell Limit com parâmetros determinados
SellStop	Coloca a ordem pendente do tipo Sell Stop com parâmetros determinados
Acesso aos últimos parâmetros dos pedidos	
Request	Obtém a cópia da última estrutura de pedido
RequestAction	Obtém o tipo de operação de negociação
RequestActionDescription	Obtém o tipo de operação de negociação como string
RequestMagic	Obtém o número mágico do Expert Advisor
RequestOrder	Obtém o ticket de ordem, utilizado no último pedido
RequestSymbol	Obtém o nome do símbolo, utilizado no último pedido
RequestVolume	Obtém o volume da negociação (em lotes), utilizado no último pedido
RequestPrice	Obtém o preço, utilizado no último pedido
RequestStopLimit	Obtém o preço da ordem pendente do tipo Stop Limit, utilizado no último pedido
RequestSL	Obtém o preço do Stop Loss da ordem, utilizado no último pedido
RequestTP	Obtém o preço Lucro do Take Profit, utilizado no último pedido
RequestDeviation	Obtém o desvio de preço da ordem, utilizado no último pedido
RequestType	Obtém o tipo de ordem, utilizado no último pedido

© 2000-2024, MetaQuotes Ltd.

Definição dos parâmetros	
RequestTypeDescription	Obtém o tipo de ordem (como string), utilizado no último pedido
RequestTypeFilling	Obtém o tipo de preenchimento da ordem, utilizado no último pedido
RequestTypeFillingDescription	Obtém o tipo de preenchimento da ordem (como string), utilizado no último pedido
RequestTypeTime	Obtém o período de validade da ordem, utilizado no último pedido
RequestTypeTimeDescription	Obtém o período de validade da ordem (como string), utilizado no último pedido
RequestExpiration	Obtém o tempo de validade da ordem, utilizado no último pedido
RequestComment	Obtém o comentário da ordem, utilizado no último pedido
RequestPosition	Obtém os bilhetes da posição.
RequestPositionBy	Obtém os bilhetes da posição oposta.
Acesso aos resultados de verificação do último pedido	
CheckResult	Obtém a cópia da estrutura do resultado da verificação do último pedido.
CheckResultRetcode	Obtém o valor do campo retcode do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultRetcodeDescription	Obtém a string da descrição do campo retcode do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultBalance	Obtém o valor do campo saldo do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultEquity	Obtém o valor do campo capital líquido do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultProfit	Obtém o valor do campo lucro do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultMargin	Obtém o valor do campo margem do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultMarginFree	Obtém o valor do campo margin_free do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
CheckResultMarginLevel	Obtém o valor do campo margin_level do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido

© 2000-2024, MetaQuotes Ltd.

Definição dos parâmetros	
CheckResultComment	Obtém o valor do campo comentário do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido
Acesso aos resultados da execução do último pedido	
Result	Obtém a cópia da estrutura do resultado do último pedido
ResultRetcode	Obtém o código de resultado do pedido
ResultRetcodeDescription	Obtém o código de resultado do pedido como texto
ResultDeal	Obtém o ticket da oferta
ResultOrder	Obtém o ticket da ordem
ResultVolume	Obtém o volume da oferta ou ordem
ResultPrice	Obtém o preço, confirmado pela corretora
ResultBid	Obtém o preço bid atual (requote)
ResultAsk	Obtém o preço ask atual (requote)
ResultComment	Obtém o comentário da corretora
Métodos auxiliares	
PrintRequest	Imprime os últimos parâmetros do pedido no diário
PrintResult	Imprime os resultados do último pedido no diário
FormatRequest	Prepara a string formatada com parâmetros do último pedido
FormatRequestResult	Prepara a string formatada com os resultados da última execução do pedido

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

LogLevel

Define nível de entrada para mensagens.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // logging level
   )
```

Parâmetros

log_level

[in] Nível de entrada.

Valor de retorno

Nenhum.

Observação

LOG_LEVEL_NO desabilita a exibição de todas as mensagens (configurado automaticamente no modo otimização). LOG_LEVEL_ERRORS permite exibir apenas mensagens de erro (valor por padrão). LOG_LEVEL_ALL permite exibir qualquer mensagem (configurado automaticamente no modo de teste).

ENUM_LOG_LEVELS

Identificador	Descrição	Valor
LOG_LEVEL_NO	Exibindo as mensagens desativadas	0
LOG_LEVEL_ERRORS	Somente as mensagens de erro são exibidas	1
LOG_LEVEL_ALL	Todas as mensagens são exibidas	2

© 2000-2024, MetaQuotes Ltd.

SetExpertMagicNumber

Define a ID do expert.

```
void  SetExpertMagicNumber(
   ulong  magic      // ID
   )
```

Parâmetros

magic

[in] Nova ID do expert.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

SetDeviationInPoints

Define o desvio permitido.

```
void  SetDeviationInPoints(
   ulong  deviation      // deviation
   )
```

Parâmetros

deviation

[in] Desvio permitido.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

SetTypeFilling

Define o tipo de preenchimento da ordem.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // order filling type
   )
```

Parâmetros

filling

[in] tipo de preenchimento da ordem (valor do enumerador ENUM_ORDER_TYPE_FILLING).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

SetTypeFillingBySymbol

Define o tipo de ordem segundo a execução de acordo com as configurações do símbolo especificado.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // nome do ativo
   )
```

Parâmetros

symbol

[in] Nome do símbolo no qual SYMBOL_FILLING_MODE contém as políticas válidas para a execução de ordens.

Valor de retorno

true - se for bem-sucedida, false - se não for possível definir a política de execução.

Observação

Se, para o símbolo, forem permitidas simultaneamente as políticas de execução SYMBOL_FILLING_FOK e SYMBOL_FILLING_IOC, para a ordem será definido o valor ORDER_FILLING_FOK.

© 2000-2024, MetaQuotes Ltd.

SetAsyncMode

Define o modo assíncrono para operações de negociação.

```
void  SetAsyncMode(
   bool  mode      // asynchronous mode flag
   )
```

Parâmetros

mode

[in] Flag de modo assíncrono.

Valor de retorno

Nenhum.

Observação

Este modo é usado para operações de negociação assíncrona, sem esperar pela resposta do servidor de negociação a um pedido enviado (ver OrderSendAsync).

© 2000-2024, MetaQuotes Ltd.

SetMarginMode

Define o modo de cálculo da margem de acordo com as configurações da conta atual.

```
void  SetMarginMode()
```

valor de retorno

Não.

Observação

O modo de cálculo da margem é indicado no ENUM_ACCOUNT_MARGIN_MODE.

© 2000-2024, MetaQuotes Ltd.

OrderOpen

Coloca a ordem pendente com parâmetros definidos.

```
bool  OrderOpen(
   const string          symbol,          // symbol
   ENUM_ORDER_TYPE       order_type,      // order type
   double                volume,          // order volume
   double                limit_price,     // StopLimit price
   double                price,           // execution price
   double                sl,              // Stop Loss price
   double                tp,              // Take Profit price
   ENUM_ORDER_TYPE_TIME  type_time,       // type by expiration
   datetime              expiration,      // expiration
   const string          comment=""       // comment
   )
```

Parâmetros

symbol

[in] Nome do instrumento de negociação.

order_type

[in] Tipo de ordem da operação de negociação (valor do enumerador ENUM_ORDER_TYPE).

volume

[in] Solicita o volume da ordem.

limit_price

[in] Preço que a ordem StopLimit será colocada.

price

[in] Preço que a ordem deve ser executada.

sl

[in] Preço que o Stop Loss será alcançado.

tp

[in] Preço que o Take Profit será alcançado.

type_time

[in] Tipo de ordem pela execução (valor do enumerador ENUM_ORDER_TYPE_TIME).

expiration

[in] A data de expiração da ordem pendente.

comment=""

[in] Comentário da ordem

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Observação

A conclusão bem sucedida do método OrderSend(...) nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação de negociação (código de retorno do servidor de negociação), utilizando ResultRetcode() e o valor, retorno por ResultOrder().

© 2000-2024, MetaQuotes Ltd.

OrderModify

Modifica os parâmetros de ordem pendentes.

```
bool  OrderModify(
   ulong                 ticket,         // order ticket
   double                price,          // execution price
   double                sl,             // Stop Loss price
   double                tp,             // Take Profit price
   ENUM_ORDER_TYPE_TIME  type_time,      // type by expiration
   datetime              expiration,     // expiration
   double                stoplimit       // Limit order price
   )
```

Parâmetros

ticket

[in] Ticket da Ordem

price

[in] O novo preço para executar a ordem (ou o valor anterior, se a mudança não for necessária).

sl

[in] O novo preço para executar o Stop Loss (ou o valor anterior, se a mudança não for necessária).

tp

[in] O novo preço para executar o Take Profit (ou o valor anterior, se a mudança não for necessária).

type_time

[in] O novo tipo de ordem de expiração (ou o valor anterior, se a mudança não for necessário), o valor do enumerador ENUM_ORDER_TYPE_TIME.

expiration

[in] A nova data de expiração de ordem pendente (ou o valor anterior, se a mudança não for necessária).

stoplimit

[in] Novo preço usado para definir uma ordem Limit, quando o preço alcançar valor do parâmetro *price*. É especificado somente para ordens do tipo StopLimit.

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

Observação

A conclusão bem sucedida do método OrderModify(...) nem sempre significa a execução bem sucedida operação de negociação. É necessário verificar o resultado do pedido de negociação (código de retorno do servidor de negociação) utilizando ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

OrderDelete

Exclui a ordem pendente.

```
bool  OrderDelete(
   ulong  ticket      // order ticket
   )
```

Parâmetros

ticket

[in] Ticket da Ordem

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

Observação

A conclusão bem sucedida do método OrderDelete(...) nem sempre significa execução bem sucedida da operação de negociação. É necessário verificar o resultado do pedido de negociação (código de retorno do servidor de negociação) utilizando ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

PositionOpen

Abre a posição com parâmetros definidos.

```
bool  PositionOpen(
   const string     symbol,          // symbol
   ENUM_ORDER_TYPE  order_type,      // order type to open position
   double           volume,          // position volume
   double           price,           // execution price
   double           sl,              // Stop Loss price
   double           tp,              // Take Profit price
   const string     comment=""       // comment
   )
```

Parâmetros

symbol

[in] Nome do instrumento de negociação para a posição aberta.

order_type

[in] Tipo de ordem (operação de negociação) para a posição aberta (valor do enumerador ENUM_ORDER_TYPE).

volume

[in] Volume da posição solicitada.

price

[in] Preço em que a posição deve ser aberta.

sl

[in] Preço que o Stop Loss será alcançado.

tp

[in] Preço que o Take Profit será alcançado.

comment=""

[in] Posição do comentário.

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

Observação

A conclusão bem sucedida do método PositionOpen(...) nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação de negociação (código de retorno do servidor da negociação), utilizando ResultRetcode() e o valor, retornado por ResultDeal().

© 2000-2024, MetaQuotes Ltd.

PositionModify

Modifica os parâmetros de posição pelo símbolo determinado.

```
bool  PositionModify(
   const string  symbol,      // símbolo
   double        sl,          // preço Stop Loss
   double        tp           // preço Take Profit
   )
```

Altera os parâmetros da posição segundo o bilhete indicado.

```
bool  PositionModify(
   const ulong   ticket,      // bilhete da posição
   double        sl,          // preço Stop Loss
   double        tp           // preço Take Profit
   )
```

Parâmetros

symbol

[in] Nome do instrumento de negociação para modificar a posição.

ticket

[In] Bilhete da posição que deverá ser modificada.

sl

[in] O novo preço para executar o Stop Loss (ou o valor anterior, se a mudança não for necessária).

tp

[in] O novo preço para executar o Take Profit (ou o valor anterior, se a mudança não for necessária).

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

Observação

A conclusão bem sucedida do método PositionModify(...) nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado do pedido de negociação (código de retorno do servidor de negociação) utilizando ResultRetcode().

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições. Neste casi, a PositionModify altera a posição com o menor bilhete.

© 2000-2024, MetaQuotes Ltd.

PositionClose

Fecha a posição pelo símbolo deteminado

```
bool  PositionClose(
   const string  symbol,                  // símbolo
   ulong         deviation=ULONG_MAX      // desvio
   )
```

Fecha a posição com o bilhete indicado.

```
bool  PositionClose(
   const ulong   ticket,                  //  bilhete da posição
   ulong         deviation=ULONG_MAX      // desvio
   )
```

Parâmetros

symbol

[in] Nome do instrumento de negociação para fechar posição.

ticket

[in] Bilhete da posição fechada.

deviation=ULONG_MAX

[in] Desvio máximo do preço atual (em pontos).

Valor de retorno

verdadeiro - em caso de verificação bem sucedida das estruturas básicas, caso contrário - falso.

Observação

Concluído com sucesso pelo método PositionClose(...), mas nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado do pedido de negociação (código de retorno do servidor de negociação) utilizando ResultRetcode().

Ao usar o registro de posições com "cobertura" (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições. Neste casi, a PositionClose fecha a posição com o menor bilhete.

© 2000-2024, MetaQuotes Ltd.

PositionClosePartial

Fecha parte da posição segundo o símbolo especificado, ao estar ativa a contabilidade de cobertura.

```
bool  PositionClosePartial(
   const string  symbol,                  // símbolo
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // desvio
   )
```

Fecha parte da posição com o bilhete indicado, ao estar ativa a contabilidade de cobertura.

```
bool  PositionClosePartial(
   const ulong   ticket,                  // bilhete da posição
   const double  volume,                  // volume
   ulong         deviation=ULONG_MAX      // desvio
   )
```

Parâmetros

symbol

[in] Nome do instrumento de negociação, segundo o qual deverá ser fechada parte da posição. Se, para o fechamento parcial da posição, for especificado o símbolo (sem bilhete), será selecionada a primeira posição encontrado nesse símbolo (identificador do Expert Advisor). Por conseguinte, em alguns casos, é melhor usar a varante PositionClosePartial() com indicação do bilhete da posição.

volume

[in] Volume para o qual é necessário reduzir a posição. Se o valor for maior do que o volume da posição parcialmente fechada, a posição será fechada completamente. Não irá ser aberta uma nova posição na direção oposta.

ticket

[in] Bilhete da posição fechada.

deviation=ULONG_MAX

[in] Desvio máximo a partir do preço atual (em pontos).

Valor de retorno

true - no caso de uma verificação básica bem-sucedida, caso contrário, false.

Observação

A conclusão bem-sucedida do método PositionClosePartial(...) nem sempre indica a conclusão bem-sucedida da operação de negociação. É necessário verificar o resultado da solicitação de negociação (código de retorno do servidor de negocião) chamando usando o método ResultRetcode().

Ao usar a contabilidade de cobertura para posições (ACCOUNT_MARGIN_MODE_RETAIL_NETTING e ACCOUNT_MARGIN_MODE_EXCHANGE) segundo cada símbolo, a qualquer momento, pode estar aberta apenas a posição que seja o resultado de uma o mais transações. Não confunda as posições e as ordens pendentes ativas que são exibidas na guia "Negociação" na janela "Caixa de ferramentas".

© 2000-2024, MetaQuotes Ltd.

Ao usar a exibição independente de posições, (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) segundo cada símbolo, podem ao mesmo tempo existir várias posições. Neste casi, a PositionClose fecha a posição com o menor bilhete.

© 2000-2024, MetaQuotes Ltd.

PositionCloseBy

Fecha a posição com o bilhete indicado usando a posição com direção oposta.

```
bool  PositionCloseBy(
   const ulong   ticket,        // bilhete da posição
   const ulong   ticket_by      // bilhete da posição oposta
   )
```

Parâmetros

ticket

[in] Bilhete da posição fechada.

ticket_by

[in] Bilhete da posição oposta usada para o fechamento.

valor de retorno

true - no caso de uma verificação básica bem-sucedida, caso contrário, false.

Observação

A conclusão bem-sucedida do método PositionCloseBy(...) nem sempre indica a conclusão bem-sucedida da operação de negociação. É necessário verificar o resultado da solicitação de negociação (código de retorno do servidor de negocião) chamando usando o método ResultRetcode().

© 2000-2024, MetaQuotes Ltd.

Buy

Abre uma posição comprada com parâmetros determinados.

```
bool  Buy(
   double        volume,          // position volume
   const string  symbol=NULL,     // symbol
   double        price=0.0,       // price
   double        sl=0.0,          // stop loss price
   double        tp=0.0,          // take profit price
   const string  comment=""       // comment
   )
```

Parâmetros

volume

[in] Volume da posição.

symbol=NULL

[in] Posição do símbolo. Se o símbolo não for especificado, o símbolo atual será usado.

price=0.0

[in] Preço. Se o preço não for especificado, o mercado usará o preço Ask.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

comment=""

[in] Comentário.

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método Buy(...) nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultDeal().

© 2000-2024, MetaQuotes Ltd.

Sell

Abre uma posição vendida com os parâmetros determinados.

```
bool  Sell(
   double        volume,          // position volume
   const string  symbol=NULL,     // symbol
   double        price=0.0,       // price
   double        sl=0.0,          // stop loss price
   double        tp=0.0,          // take profit price
   const string  comment=""       // comment
   )
```

Parâmetros

volume

[in] Volume da posição.

symbol=NULL

[in] Posição do símbolo. Se o símbolo não for especificado, o símbolo atual será usado.

price=0.0

[in] Preço. Se o preço não for determinado, o preço Bid atual de mercado será usado.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

comment=""

[in] Comentário.

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método Sell(...) nem sempre significa execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultDeal().

© 2000-2024, MetaQuotes Ltd.

BuyLimit

Coloca a ordem pendente do tipo Buy Limit (compra no preço menor do que o preço atual de mercado) com parâmetros determinados .

```
bool  BuyLimit(
   double                volume,                      // order volume
   double                price,                       // order price
   const string          symbol=NULL,                 // symbol
   double                sl=0.0,                      // stop loss price
   double                tp=0.0,                      // take profit price
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // order lifetime
   datetime              expiration=0,                // order expiration time
   const string          comment=""                   // comment
   )
```

Parâmetros

volume

[in] Volume da ordem.

price

[in] Preço da ordem.

symbol=NULL

[in] Símbolo da ordem Se o símbolo não for especificado, o símbolo atual será usado.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

type_time=ORDER_TIME_GTC

[in] Ordem infinita (valor do enumerador ENUM_ORDER_TYPE_TIME).

expiration=0

[In] Tempo de expiração da ordem (usado somente se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentário da ordem

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método BuyLimit(...) nem sempre significa a execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultOrder().

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

BuyStop

Coloca a ordem pendente do tipo Buy Stop (compra no preço maior do que o preço atual de mercado) com parâmetros determinados.

```
bool  BuyStop(
   double                volume,                      // order volume
   double                price,                       // order price
   const string          symbol=NULL,                 // symbol
   double                sl=0.0,                      // stop loss price
   double                tp=0.0,                      // take profit price
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // order lifetime
   datetime              expiration=0,                // order expiration time
   const string          comment=""                   // comment
   )
```

Parâmetros

volume

[in] Volume da ordem.

price

[in] Preço da ordem.

symbol=NULL

[in] Símbolo da ordem Se o símbolo não for especificado, o símbolo atual será usado.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

type_time=ORDER_TIME_GTC

[in] Ordem infinita (valor do enumerador ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Tempo de expiração da ordem (usado somente se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentário da ordem

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método BuyStop(...) nem sempre significa execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultOrder().

© 2000-2024, MetaQuotes Ltd.

SellLimit

Coloca a ordem pendente do tipo Sell Limit (vender pelo preço maior do que o preço atual de mercado) com parâmetros determinados.

```
bool  SellLimit(
   double                volume,                      // order volume
   double                price,                       // order price
   const string          symbol=NULL,                 // symbol
   double                sl=0.0,                      // stop loss price
   double                tp=0.0,                      // take profit price
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // order lifetime
   datetime              expiration=0,                // order expiration time
   const string          comment=""                   // comment
   )
```

Parâmetros

volume

[in] Volume da ordem.

price

[in] Preço da ordem.

symbol=NULL

[in] Símbolo da ordem Se o símbolo não for especificado, o símbolo atual será usado.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

type_time=ORDER_TIME_GTC

[in] Ordem infinita (valor do enumerador ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Tempo de expiração da ordem (usado somente se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentário da ordem

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método SellLimit(...) nem sempre significa execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultOrder().

© 2000-2024, MetaQuotes Ltd.

SellStop

Coloca a ordem pendente do tipo Sell Stop (vender pelo preço mais baixo do que o preço atual de mercado) com parâmetros determinados.

```
bool  SellStop(
   double                volume,                      // order volume
   double                price,                       // order price
   const string          symbol=NULL,                 // symbol
   double                sl=0.0,                      // stop loss price
   double                tp=0.0,                      // take profit price
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,    // order lifetime
   datetime              expiration=0,                // order expiration time
   const string          comment=""                   // comment
   )
```

Parâmetros

volume

[in] Volume da ordem.

price

[in] Preço da ordem.

symbol=NULL

[in] Símbolo da ordem Se o símbolo não for especificado, o símbolo atual será usado.

sl=0.0

[in] preço para finalizar a perda do capital (Stop Loss).

tp=0.0

[in] Preço do Take Profit.

type_time=ORDER_TIME_GTC

[in] Ordem infinita (valor do enumerador ENUM_ORDER_TYPE_TIME).

expiration=0

[in] Tempo de expiração da ordem (usado somente se type_time=ORDER_TIME_SPECIFIED).

comment=""

[in] Comentário da ordem

Valor de retorno

verdadeiro - no caso do sucesso da verificação das estruturas, caso contrário - falso.

Observação

A conclusão bem sucedida do método SellStop(...) nem sempre significa execução bem sucedida da operação de negociação. É necessário verificar o resultado da solicitação da negociação (código de retorno do servidor de negociação) usando o ResultRetcode() e o valor, retorno pelo ResultOrder().

© 2000-2024, MetaQuotes Ltd.

Request

Obtém a cópia da última estrutura de pedido.

```
void  Request(
   MqlTradeRequest&  request      // target structure
   ) const
```

Parâmetros

request

[Out] Referente a estrutura do tipo MqlTradeRequest.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

RequestAction

Obtém o tipo de operação de negociação.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Valor de retorno

Tipo de operação de negociação, usando o último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestActionDescription

Obtém o tipo de operação de negociação como string.

```
string  RequestActionDescription() const
```

Valor de retorno

Tipo de operação de negociação (como string), utilizada no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestMagic

Obtém o número mágico do Expert Advisor.

```
ulong  RequestMagic() const
```

Valor de retorno

O número mágico (ID) do Expert Advisor, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestOrder

Obtém o ticket de ordem, utilizado no último pedido.

```
ulong  RequestOrder() const
```

Valor de retorno

Ticket de ordem do último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestSymbol

Obtém o nome do símbolo, utilizado no último pedido.

```
string  RequestSymbol() const
```

Valor de retorno

O nome do símbolo, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestVolume

Obtém o volume de negociação (em lotes), utilizado no último pedido.

```
double  RequestVolume() const
```

Valor de retorno

O volume da negociação (em lotes), utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestPrice

Obtém o preço, usado no último pedido.

```
double  RequestPrice() const
```

Valor de retorno

Preço da ordem, usado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestStopLimit

Obtém o preço da ordem pendente do tipo Stop Limit, utilizado no último pedido.

```
double  RequestStoplimit() const
```

Valor de retorno

O preço da ordem pendente do tipo Stop Limit, no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestSL

Obtém o preço do Stop Loss da ordem, utilizado no último pedido.

```
double  RequestSL() const
```

Valor de retorno

O preço do Stop Loss, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestTP

Obtém o preço do Take Profit da ordem, utilizado no último pedido.

```
double  RequestTP() const
```

Valor de retorno

O preço do Take Profit , usado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestDeviation

Obtém o desvio de preço da ordem, utilizado no último pedido.

```
ulong  RequestDeviation() const
```

Valor de retorno

O desvio de preço da ordem, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestType

Obtém o tipo de ordem, utilizado no último pedido.

```
ENUM_ORDER_TYPE  RequestType() const
```

Valor de retorno

Tipo de ordem, utilizado no último pedido (valor do enumerador ENUM_ORDER_TYPE).

© 2000-2024, MetaQuotes Ltd.

RequestTypeDescription

Obtém o tipo de ordem (como string), utilizado no último pedido.

```
string  RequestTypeDescription() const
```

Valor de retorno

O tipo de ordem (como string), utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestTypeFilling

Obtém do tipo de preenchimento da ordem, utilizado no último pedido.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Valor de retorno

O tipo de preenchimento da ordem (valor do enumerador ENUM_ORDER_TYPE_FILLING), utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestTypeFillingDescription

Obtém o tipo de preenchimento da ordem (como string), utilizado no último pedido.

```
string  RequestTypeFillingDescription() const
```

Valor de retorno

O tipo de preenchimento (como string) da ordem, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestTypeTime

Obtém o período de validade da ordem, utilizado no último pedido.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Valor de retorno

O período de validade da ordem (valor do enumerador ENUM_ORDER_TYPE_TIME), utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestTypeTimeDescription

Obtém o período de validade da ordem (como string), utilizado no último pedido.

```
string  RequestTypeTimeDescription() const
```

Valor de retorno

O período de validade da ordem (como string), utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestExpiration

Obtém o tempo de validade da ordem, utilizado no último pedido.

```
datetime  RequestExpiration() const
```

Valor de retorno

O tempo de validade da ordem, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestComment

Obtém o comentário da ordem, utilizado no último pedido.

```
string  RequestComment() const
```

Valor de retorno

O comentário da ordem, utilizado no último pedido.

© 2000-2024, MetaQuotes Ltd.

RequestPosition

Obtém o bilhete da posição.

```
ulong  RequestPosition() const
```

valor de retorno

Bilhete da posição usada na última solicitação.

© 2000-2024, MetaQuotes Ltd.

RequestPositionBy

Obtém o bilhete da posição oposta.

```
ulong  RequestPositionBy() const
```

valor de retorno

Bilhete da posição oposta usada na última solicitação.

© 2000-2024, MetaQuotes Ltd.

Result

Obtém a cópia da estrutura do resultado do último pedido.

```
void  Result(
   MqlTradeResult&  result      // reference
   ) const
```

Parâmetros

result

[Out] Referente a estrutura do tipo MqlTradeResult.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

ResultRetcode

Obtém o código de resultado do pedido.

```
uint  ResultRetcode() const
```

Valor de retorno

O código do resultado do pedido.

© 2000-2024, MetaQuotes Ltd.

ResultRetcodeDescription

Obtém o código de resultado do pedido como texto.

```
string  ResultRetcodeDescription() const
```

Valor de retorno

Código de resultado do pedido como texto.

© 2000-2024, MetaQuotes Ltd.

ResultDeal

Obtém o ticket da oferta.

```
ulong  ResultDeal() const
```

Valor de retorno

Ticket da oferta, se o negócio foi executado.

© 2000-2024, MetaQuotes Ltd.

ResultOrder

Obtém o ticket da ordem.

```
ulong  ResultOrder() const
```

Valor de retorno

Ticket da ordem, se a ordem foi colocada.

© 2000-2024, MetaQuotes Ltd.

ResultVolume

Obtém o volume de oferta ou ordem.

```
double ResultVolume() const
```

Valor de retorno

Volume de oferta ou ordem.

© 2000-2024, MetaQuotes Ltd.

ResultPrice

Obtém o preço, confirmado pela corretora.

```
double ResultPrice() const
```

Valor de retorno

Preço, confirmado pela corretora.

© 2000-2024, MetaQuotes Ltd.

ResultBid

Obtém o preço bid atual (requote).

```
double  ResultBid() const
```

Valor de retorno

Preço bid atual (requote).

© 2000-2024, MetaQuotes Ltd.

ResultAsk

Obtém o preço ask atual (requote).

```
double  ResultAsk() const
```

Valor de retorno

Preço ask atual (requote).

© 2000-2024, MetaQuotes Ltd.

ResultComment

Obtém o comentário da corretora.

```
string  ResultComment() const
```

Valor de retorno

Comentário da corretora sobre a operação.

© 2000-2024, MetaQuotes Ltd.

CheckResult

Obtém a cópia da estrutura do resultado da verificação do último pedido.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // reference
   ) const
```

Parâmetros

check_result

[out] Referente a estrutura de destino do tipo MqlTradeCheckResult.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CheckResultRetcode

Obtém o valor do campo retcode do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
uint  CheckResultRetcode() const
```

Valor de retorno

O valor do campo retcode (código de erro) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultRetcodeDescription

Obtém a string de descrição do campo retcode do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
string  ResultRetcodeDescription() const
```

Valor de retorno

A descrição da string do campo retcode (código de erro) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultBalance

Obtém o valor do campo saldo do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double  CheckResultBalance() const
```

Valor de retorno

O valor do campo saldo (valor do saldo após a execução da operação de negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultEquity

Obtém o valor do campo capital líquido do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double  CheckResultEquity() const
```

Valor de retorno

O valor do campo capital líquido (valor do capital líquido será após a execução da operação de negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultProfit

Obtém o valor do campo lucro do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double  CheckResultProfit() const
```

Valor de retorno

O valor do campo de lucro (o valor lucro será após a execução da operação de negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultMargin

Obtém o valor do campo margem do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double CheckResultMargin() const
```

Valor de retorno

O valor do campo margem (margem necessária para o funcionamento da negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultMarginFree

Obtém o valor do campo margin_free do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double  CheckResultMarginFree() const
```

Valor de retorno

O valor do campo margin_free (margem livre que ficará após a execução da operação de negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultMarginLevel

Obtém o valor do campo margin_level do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
double  CheckResultMarginLevel() const
```

Valor de retorno

O valor do campo margin_level (nível de margem será ajustado após a execução da operação de negociação) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

CheckResultComment

O valor do campo de comentário do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

```
string  CheckResultComment() const
```

Valor de retorno

O valor do campo comentário (comentário ao código de resposta, descrição de erro) do tipo MqlTradeCheckResult, preenchido durante a verificação da exatidão do pedido.

© 2000-2024, MetaQuotes Ltd.

PrintRequest

Imprime os últimos parâmetros do pedido no diário.

```
void PrintRequest() const
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

PrintResult

Imprime os resultados do último pedido no diário.

```
void PrintResult() const
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

FormatRequest

Prepara a string formatada com parâmetros do último pedido.

```
string  FormatRequest(
   string&                 str,          // target string
   const MqlTradeRequest&  request       // request
   ) const
```

Parâmetros

str

[in] String de destino, transmitida por referência.

request

[A] Uma estrutura do tipo MqlTradeRequest com parâmetros do último pedido.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

FormatRequestResult

Prepara a string formatada com os resultados da última execução do pedido.

```
string  FormatRequestResult(
   string&                 str,          // string
   const MqlTradeRequest&  request,      // request structure
   const MqlTradeResult&   result        // result structure
   ) const
```

Parâmetros

str

[in] String de destino, transmitida por referência.

request

[in] Uma estrutura do tipo MqlTradeRequest com parâmetros do último pedido.

result

[in] Uma estrutura do tipo MqlTradeResult com os resultados da último pedido.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CTerminalInfo

CTerminalInfo é uma classe de acesso simplificado às propriedades do ambiente do programa MQL5.

Descrição

A classe CTerminalInfo possibilita o acesso às propriedades do ambiente do programa MQL5.

Declaração

```
class CTerminalInfo : public CObject
```

Título

```
#include <Trade\TerminalInfo.mqh>
```

Hierarquia de herança

CObject
  CTerminalInfo

Métodos de classe pelos grupos

Métodos de acesso às propriedades do tipo integer	
Build	Obtém o número de compilação do terminal do cliente
IsConnected	Obtém as informações sobre a conexão no servidor de negociação
IsDLLsAllowed	Obtém as informações sobre a permissão de uso de arquivo DLL
IsTradeAllowed	Obtém as informações sobre a permissão de negociação
IsEmailEnabled	Obtém as informações sobre a permissão para enviar e-mails para servidor SMTP e login, especificado nas configurações do terminal
IsFtpEnabled	Obtém as informações sobre a permissão para enviar relatórios de negociação ao servidor FTP e login, especificado nas configurações do terminal
MaxBars	Obtém as informações sobre o número máximo de barras no gráfico
CodePage	Obtém as informações sobre a página de código do idioma no terminal do cliente
CPUCores	Obtém as informações sobre os núcleos da CPU
MemoryPhysical	Obtém as informações sobre a memória física (em Mb)
MemoryTotal	Obtém as informações sobre o total de memória, disponível para processamento do terminal/agente (em Mb)

© 2000-2024, MetaQuotes Ltd.

Métodos de acesso às propriedades do tipo integer	
MemoryAvailable	Obtém as informações sobre a memória livre, disponível para processamento do terminal/agente (em Mb)
MemoryUsed	Obtém as informações sobre a memória, utilizada pelo processamento do terminal/agente (em Mb)
IsX64	Obtém as informações sobre o tipo de terminal do cliente (32/64 bit)
OpenCLSupport	Obtém as informações sobre a versão do OpenCL, apoiada pela placa de vídeo
DiskSpace	Obtém as informações sobre o espaço livre em disco (em Mb)
Métodos para acesso às propriedades do tipo string	
Language	Obtém o idioma do terminal do cliente
Name	Obtém o nome do terminal do cliente
Company	Obtém a empresa do terminal do cliente
Path	Obtém a pasta do terminal de cliente
DataPath	Obtém a pasta de dados do terminal do cliente
CommonDataPath	Obtém a pasta de dados comum de todos os terminais de cliente, instalados no computador
Acesso a funções API MQL5	
InfoInteger	Obtém o valor da propriedade de tipo inteiro
InfoString	Obtém o valor da propriedade do tipo string

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Build

Obtém o número de compilação do terminal do cliente.

```
int  CBuild() const
```

Valor de retorno

Número de compilação do terminal do cliente.

Observação

Para obter o número de compilação que usa a função TerminalInfoInteger() (propriedade TERMINAL_BUILD).

© 2000-2024, MetaQuotes Ltd.

IsConnected

Obtém as informações sobre a conexão no servidor de negociação.

```
bool  IsConnected() const
```

Valor de retorno

verdadeiro, se o terminal está conectado no servidor de negociação, caso contrário é falso.

Observação

Para obter o status da conexão, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_CONNECTED).

© 2000-2024, MetaQuotes Ltd.

IsDLLsAllowed

Obtém as informações sobre a permissão de uso de arquivo DLL.

```
bool  IsDLLsAllowed() const
```

Valor de retorno

verdadeiro, se o uso DLL foi permitido, caso contrário é falso.

Observação

Para obter a permissão de uso de DLL, usa a função TerminalInfoInteger() (propriedade do TERMINAL_DLLS_ALLOWED).

© 2000-2024, MetaQuotes Ltd.

IsTradeAllowed

Obtém as informações sobre a permissão de negociação.

```
bool  IsTradeAllowed() const
```

Valor de retorno

verdadeiro, se a negociação foi permitida, caso contrário é falso.

Observação

Para obter permissão de negociação, usa a função TerminalInfoInteger() (propriedade TERMINAL_TRADE_ALLOWED).

© 2000-2024, MetaQuotes Ltd.

IsEmailEnabled

Obtém as informações sobre a permissão para enviar e-mails para o servidor SMTP e login, especificado nas configurações do terminal.

```
bool  IsEmailEnabled() const
```

Valor de retorno

verdadeiro, se o envio de e-mails foi permitido, caso contrário é falso.

Observação

Para obter a permissão para enviar e-mails, usa a função TerminalInfoInteger() (propriedade TERMINAL_EMAIL_ENABLED).

© 2000-2024, MetaQuotes Ltd.

IsFtpEnabled

Obtém as informações sobre a permissão para enviar relatórios de negociação ao servidor FTP e login, especificado nas configurações do terminal.

```
bool  IsFtpEnabled() const
```

Valor de retorno

verdadeiro, se os envio de relatórios de negociação para o servidor FTP foi permitido, caso contrário é falso.

Observação

Para obter as informações sobre a permissão para enviar relatórios de negociação, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_FTP_ENABLED).

© 2000-2024, MetaQuotes Ltd.

MaxBars

Obtém o número máximo de barras no gráfico, especificado nas configurações do terminal cliente.

```
int  MaxBars() const
```

Valor de retorno

O número máximo de barras no gráfico.

Observação

Para obter o número máximo de barras no gráfico, usa a função TerminalInfoInteger() (propriedade TERMINAL_MAXBARS).

© 2000-2024, MetaQuotes Ltd.

CodePage

Obtém as informações sobre a página de código do idioma no terminal do cliente.

```
int  CodePage() const
```

Valor de retorno

Página de código do idioma no terminal do cliente.

Observação

Para obter a página de código que utiliza a função TerminalInfoInteger() (propriedade TERMINAL_CODEPAGE).

© 2000-2024, MetaQuotes Ltd.

CPUCores

Obtém as informações sobre a quantidade de núcleos da CPU no sistema.

```
int  CPUCores() const
```

Valor de retorno

Quantidade de núcleos da CPU no sistema.

Observação

Para obter a quantidade de núcleos de CPU, usa a função TerminalInfoInteger() (propriedadeTERMINAL_CPU_CORES).

© 2000-2024, MetaQuotes Ltd.

MemoryPhysical

Obtém as informações sobre a memória física (em Mb).

```
int  MemoryPhysical() const
```

Valor de retorno

Memória física (em Mb).

Observação

Para obter a memória física, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_MEMORY_PHYSICAL).

© 2000-2024, MetaQuotes Ltd.

MemoryTotal

Obtém as informações sobre o total de memória, disponível para processamento do terminal/agente (em Mb).

```
int  MemoryTotal() const
```

Valor de retorno

Total de memória (em Mb), disponível ao terminal/agente.

Observação

Para obter o total de memória, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_MEMORY_TOTAL).

© 2000-2024, MetaQuotes Ltd.

MemoryAvailable

Obtém as informações sobre a memória livre, disponível para processamento do terminal/agente (em Mb).

```
int  MemoryTotal() const
```

Valor de retorno

Memória livre (em Mb), disponível para o terminal/agente.

Observação

Para obter a memória livre, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_MEMORY_TOTAL).

© 2000-2024, MetaQuotes Ltd.

MemoryUsed

Obtém as informações sobre a memória, utilizada pelo terminal/agente (em Mb).

```
int  MemoryUsed() const
```

Valor de retorno

A memória, utilizada pelo cliente de terminal/agente (em Mb).

Observação

Para obter a memória usada pelo terminal, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_MEMORY_USED).

© 2000-2024, MetaQuotes Ltd.

IsX64

Obtém as informações sobre o tipo de terminal do cliente.

```
bool  IsX64() const
```

Valor de retorno

verdadeiro, se a versão de 64 bits está sendo usada, caso contrário é falso.

Observação

Para obter o tipo de terminal do cliente, utiliza a função TerminalInfoInteger() (propriedade TERMINAL_X64).

© 2000-2024, MetaQuotes Ltd.

OpenCLSupport

Obtém as informações sobre a versão do OpenCL, apoiada pela placa de vídeo.

```
int  OpenCLSupport() const
```

Valor de retorno

O valor retornado tem a seguinte forma: 0x00010002 = "1.2". O 0(zero) significa que não há suporte para OpenCL.

Observação

Para obter a versão do OpenCL, usa a função TerminalInfoInteger() (propriedade TERMINAL_OPENCL_SUPPORT).

© 2000-2024, MetaQuotes Ltd.

DiskSpace

Obtém as informações sobre o espaço livre em disco, disponível ao terminal/agente (em Mb).

```
int  MDiskSpace() const
```

Valor de retorno

Espaço livre em disco (em Mb), disponível ao cliente de terminal/agente (para arquivos, salvos na pasta MQL5\Files).

Observação

Para obter o espaço livre em disco, usa a função TerminalInfoInteger() (propriedade TERMINAL_DISK_SPACE).

© 2000-2024, MetaQuotes Ltd.

Language

Obtém as informações sobre o idioma do terminal do cliente.

```
string  Language() const
```

Valor de retorno

Idioma, utilizado no terminal do cliente.

Observação

Para obter o idioma, utiliza a função TerminalInfoString() (propriedade TERMINAL_LANGUAGE).

© 2000-2024, MetaQuotes Ltd.

Name

Obtém a informação do nome do terminal de cliente.

```
string  Name() const
```

Valor de retorno

Nome do terminal de cliente.

Observação

Para obter o nome do terminal do cliente, utiliza a função TerminalInfoString() (propriedade TERMINAL_NAME).

© 2000-2024, MetaQuotes Ltd.

Company

Obtém as informações sobre o nome da corretora.

```
string  Company() const
```

Valor de retorno

O nome do corretora.

Observação

Para obter o nome da corretora, usa a função TerminalInfoString() (propriedade TERMINAL_COMPANY).

© 2000-2024, MetaQuotes Ltd.

Path

Obtém a pasta do terminal de cliente.

```
string  Path() const
```

Valor de retorno

A pasta do terminal de cliente.

Observação

Para obter a pasta do terminal de cliente, utiliza a função TerminalInfoString() (propriedade TERMINAL_PATH).

© 2000-2024, MetaQuotes Ltd.

DataPath

Obtém as informações sobre a pasta de dados do terminal.

```
string  DataPath() const
```

Valor de retorno

Pasta de dados do terminal de cliente.

Observação

Para obter a pasta de dados do terminal do cliente, utiliza a função TerminalInfoString() (propriedade TERMINAL_DATA_PATH).

© 2000-2024, MetaQuotes Ltd.

CommonDataPath

Obtém a pasta de dados comum de todos os terminais de cliente, instalados no computador.

```
string  CommonDataPath() const
```

Valor de retorno

Pasta de dados comum.

Observação

Para obter pasta de dados comum que usa a função TerminalInfoString() (propriedade COMMON_DATA_PATH).

© 2000-2024, MetaQuotes Ltd.

InfoInteger

Retorna o valor de uma propriedade correspondente do ambiente de um programa mql5.

```
int  TerminalInfoInteger(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Pode ser um dos valores do enumerador ENUM_TERMINAL_INFO_INTEGER.

Valor de retorno

Valor do tipo int.

Observação

Para obter o valor da propriedade, usa a função TerminalInfoInteger().

© 2000-2024, MetaQuotes Ltd.

InfoString

A função retorna o valor de uma propriedade correspondente do ambiente do programa mql5. A propriedade deve ser do tipo string.

```
string  TerminalInfoString(
   int  property_id      // identificador de uma propriedade
   );
```

Parâmetros

property_id

[in] Identificador de uma propriedade. Pode ser um dos valores do enumerador ENUM_TERMINAL_INFO_STRING.

Valor de retorno

Valor do tipo string.

Observação

Para obter o valor da propriedade, usa a função TerminalInfoString().

© 2000-2024, MetaQuotes Ltd.

Classes de Estratégias de Negociação

Esta seção contém detalhes técnicos de como trabalhar com classes para criação, teste de estratégias de negociação e descrição dos principais componentes da biblioteca padrão MQL5.

O uso destas classes vai economizar tempo ao criar as estratégias de negociação.

A Biblioteca Padrão MQL5 (em termos de estratégias de negociação) é colocada no diretório do terminal, na pasta Include\Expert.

Classes Base	Descrição
CExpertBase	Classe base para todas as classes de estratégia de negociação
CExpert	Classe base para Expert Advisor
CExpertSignal	Classe base para classes de Trading Signal
CExpertTrailing	Classe base para classes de Trailing Stop
CExpertMoney	Classe base para classes de Money Management

Classes de sinal de negociação	Descrição
CSignalAC	O módulo de sinais baseado em modelos de mercado do indicador Accelerator Oscillator.
CSignalAMA	O módulo de sinais baseado em modelos de mercado do indicador Adaptive Moving Average.
CSignalAO	O módulo de sinais baseado em modelos de mercado do indicador Awesome Oscillator.
CSignalBearsPower	O módulo de sinais baseado em modelos de mercado do indicador oscilador Bears Power.
CSignalBullsPower	O módulo de sinais baseado em modelos de mercado do indicador oscilador Bulls Power.
CSignalCCI	O módulo de sinais baseado em modelos de mercado do indicador oscilador Commodity Channel Index.
CSignalDeM	O módulo de sinais baseado em modelos de mercado do indicador oscilador DeMarker.
CSignalDEMA	O módulo de sinais baseado em modelos de mercado do indicador Double Exponential Moving Average.
CSignalEnvelopes	O módulo de sinais baseado em modelos de mercado do indicador Envelopes.
CSignalFrAMA	O módulo de sinais baseado em modelos de mercado do indicador Fractal Adaptive Moving Average.
CSignalITF	O módulo para filtro de sinais pela hora.

© 2000-2024, MetaQuotes Ltd.

Classes de sinal de negociação	Descrição
CSignalMACD	O módulo de sinais baseado em modelos de mercado do indicador oscilador MACD.
CSignalMA	O módulo de sinais baseado em modelos de mercado do indicador Moving Average.
CSignalSAR	O módulo de sinais baseado em modelos de mercado do indicador Parabolic SAR.
CSignalRSI	O módulo de sinais baseado em modelos de mercado do indicador oscilador Relative Strength Index.
CSignalRVI	O módulo de sinais baseado em modelos de mercado do indicador oscilador Relative Vigor Index.
CSignalStoch	O módulo de sinais baseado em modelos de mercado do indicador oscilador Stochastic.
CSignalTRIX	O módulo de sinais baseado em modelos de mercado do indicador oscilador Triple Exponential Average.
CSignalTEMA	O módulo de sinais baseado em modelos de mercado do indicador Triple Exponential Moving Average.
CSignalWPR	O módulo de sinais baseado em modelos de mercado do indicador oscilador Williams Percent Range.

Classe Trailing Stop	Descrição
CTrailingFixedPips	Essa classe implementa algoritmo de Trailing Stop baseado em pontos fixos
CTrailingMA	Essa classe implementa algoritmo de Trailing Stop baseado nos valores do indicador Moving Average
CTrailingNone	Uma classe stub, não usa qualquer algoritmo de Trailing Stop
CTrailingPSAR	Essa classe implementa algoritmo de Trailing Stop baseado nos valores do indicador Parabolic SAR

Classes de Gerenciamento de Dinheiro	Descrição
CMoneyFixedLot	Uma classe com um algoritmo, baseado em negociação com tamanho predefinido de lote fixo.
CMoneyFixedMargin	Uma classe com um algoritmo, baseado em negociação com margem fixa predefinida.
CMoneyFixedRisk	Uma classe com um algoritmo, baseado em negociação com risco predefinido.

© 2000-2024, MetaQuotes Ltd.

Classes de Gerenciamento de Dinheiro	Descrição
CMoneyNone	Uma classe com um algoritmo, baseado em negociação com o tamanho do lote mínimo permitido.
CMoneySizeOptimized	Uma classe com um algoritmo, baseado em negociação com o tamanho de lote variável, em função de resultados das ofertas anteriores.

© 2000-2024, MetaQuotes Ltd.

Classes Base para Expert Advisors

Esta seção contém detalhes técnicos de como trabalhar com classes para criação, teste de estratégias de negociação e descrição dos principais componentes da biblioteca padrão MQL5.

O uso destas classes vai economizar tempo ao criar as estratégias de negociação.

A Biblioteca Padrão MQL5 (em termos de estratégias de negociação) é colocada no diretório do terminal, na pasta Include\Expert.

Classe	Descrição
CExpertBase	Classe base para todas as classes de estratégia de negociação
CExpert	Classe base para Expert Advisor
CExpertSignal	Classe base para classes de Trading Signal
CExpertTrailing	Classe base para classes de Trailing Stop
CExpertMoney	Classe base para classes de Money Management

© 2000-2024, MetaQuotes Ltd.

CExpertBase

CExpertBase é uma classe base para a CExpert e todas as classes de estratégia de negociação.

Descrição

CExpertBase fornece os dados e métodos que são comuns a todos os objetos do Expert Advisor.

Declaração

```
class CExpertBase : public CObject
```

Título

```
#include <Expert\ExpertBase.mqh>
```

Hierarquia de herança

CObject

CExpertBase

Descendentes diretos

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Métodos de classe

Métodos públicos:

Inicialização	
virtual Init	Instância de classe do método de inicialização.
virtual ValidationSettings	Verifica as configurações
Parâmetros	
Symbol	Define o símbolo
Period	Define o timeframe
Magic	Define a ID do Expert Advisor
Indicadores e TimeSeries	
virtual SetPriceSeries	Define ponteiros para timeseries externas (séries de preços)
virtual SetOtherSeries	Define ponteiros para timeseries externos (séries sem preço)
virtual InitIndicators	Inicializa os indicadores e timeseries
O acesso aos Dados Protegidos	
InitPhase	Obtém a fase atual da inicialização de objeto

© 2000-2024, MetaQuotes Ltd.

Inicialização	
TrendType	Define o tipo de tendência
UsedSeries	Obtém máscara de bits das timeseries usadas
EveryTick	Define o flag "Every tick"
Acesso as Timeseries	
Open	Obtém o elemento das timeseries "Open" pelo índice
High	Obtém o elemento das timeseries "High" pelo índice
Low	Obtém o elemento das timeseries "Low" pelo índice
Close	Obtém o elemento das timeseries "Close" pelo índice
Spread	Obtém o elemento das timeseries Spread pelo índice
Time	Obtém o elemento das timeseries Time pelo índice
TickVolume	Obtém o elemento das timeseries TickVolume pelo índice
RealVolume	Obtém o elemento das timeseries RealVolume pelo índice

Métodos Protegidos:

A inicialização da TimeSeries	
InitOpen	Método de inicialização das Timeseries Open
InitHigh	Método de inicialização das Timeseries High
InitLow	Método de inicialização das Timeseries Low
InitClose	Método de inicialização das Timeseries Close
InitSpread	Método de inicialização das Timeseries Spread
InitTime	Método de inicialização das Timeseries Time
InitTickVolume	Método de inicialização das Timeseries TickVolume
InitRealVolume	Método de inicialização das Timeseries RealVolume
Métodos de Serviços	
virtual PriceLevelUnit	Obtém a unidade do nível de preços
virtual StartIndex	Obtém o índice da barra para análise
virtual CompareMagic	Compara a ID do Expert Advisor com o valor determinado

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

InitPhase

Obtém a fase atual da inicialização de objeto.

```
ENUM_INIT_PHASE  InitPhase()
```

Valor de retorno

Fase atual da inicialização do objeto.

Observação

A inicialização do objeto consiste em várias fases:

1. Início da inicialização.

- iniciar - após finalização do construtor
- finalizar - após método Init(...) completado com sucesso.
- permitir - chamar o método Init(...)
- não permitir - chamar o método ValidationSettings() e outros métodos de incialização

2. A definição dos parâmetros da fase. Nesta fase você precisa configurar todos os parâmetros de objeto utilizados para criar indicadores.

- iniciar - após método Init(...) completado com sucesso.
- finalizar - após método ValidationSettings() completado com sucesso
- permitir - chamar os métodos Symbol(...) e Period(...)
- não permitir - chamar os métodosInit(...), SetPriceSeries(...), SetOtherSeries(...) e InitIndicators(...)

3. Verificação de parâmetros.

- iniciar - após método ValidationSettings() completado com sucesso
- finalizar - após método InitIndicators(...) completado com sucesso
- permitir - chamar os métodos Symbol(...), Period(...) e InitIndicators(...)
- não permitir - chamar qualquer método de inicialização

4. Finalização da inicialização.

- iniciar - após método InitIndicators(...) completado com sucesso
- não permitir - chamar métodos de inicialização

© 2000-2024, MetaQuotes Ltd.

TrendType

Define o tipo de tendência.

```
void  TrendType(
   M_TYPE_TREND    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do tipo tendência.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

UsedSeries

Obtém máscara de bits (bitmask) das timeseries usadas.

```
int  UsedSeries()
```

Valor de retorno

A lista das timeseries usadas como máscara de bits (bitmask).

Observação

Se o bit for definido, as timeseries correspondentes são usadas, se não for definido, as timeseries não são usadas.

As timeseries bit correspondem:

bit 0 - timeseries Open,
bit 1 - timeseries High,
bit 2 - timeseries Low,
bit 3 - timeseries Close,
bit 4 - timeseries Open,
bit 5 - timeseries Time,
bit 6 - timeseries TickVolume,
bit 7 - timeseries RealVolume.

© 2000-2024, MetaQuotes Ltd.

EveryTick

Define o flag "Every tick".

```
void  EveryTick(
   bool    value         // flag
   )
```

Parâmetros

value

[in] O novo valor da flag.

Valor de retorno

Nenhum.

Observação

Se o flag não estiver definido, o método de processamento é chamado somente no timeframe e símbolo que a nova barra está trabalhando.

© 2000-2024, MetaQuotes Ltd.

Open

Obtém o elemento das timeseries "Open" pelo índice.

```
double  Open(
   int    ind         // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries Open com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

High

Obtém o elemento das timeseries "High" pelo índice .

```
double  High(
   int    ind         // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, ele retorna o valor numérico do elemento das timeseries High pelo índice determinado, caso contrário ele retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

Low

Obtém o elemento das timeseries "Low" pelo índice.

```
double  Low(
   int    ind          // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries Low com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

Close

Obtém o elemento das timeseries "Close" pelo índice.

```
double  Close(
   int    ind         // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries Close com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

Spread

Obtém o elemento das timeseries Spread pelo índice.

```
double  Spread(
   int    ind         // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries Spread com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

Time

Obtém o elemento das timeseries Time pelo índice.

```
datetime  Time(
   int    ind          // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries Time com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

TickVolume

Obtém o elemento das timeseries TickVolume pelo índice.

```
long  TickVolume(
   int    ind         // Index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries TickVolume com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

RealVolume

Obtém o elemento das timeseries RealVolume pelo índice.

```
long  RealVolume(
   int    ind         // index
   )
```

Parâmetros

ind

[in] Índice do elemento.

Valor de retorno

Se com sucesso, retorna o valor numérico do elemento das timeseries RealVolume com índice especificado, caso contrário retorna EMPTY_VALUE.

Observação

O EMPTY_VALUE é devolvido em dois casos:

1. Timeseries não é usada (o bit correspondente não está definido).
2. Índice do elemento está fora do intervalo.

© 2000-2024, MetaQuotes Ltd.

Init

Inicializa o objeto.

```
bool  Init(
   CSymbolInfo       symbol,    // symbol
   ENUM_TIMEFRAMES   period,    // timeframe
   double            point      // point
   )
```

Parâmetros

symbol

[in] Ponteiro ao objeto de tipo CSymbolInfo para o acesso a informações de símbolo.

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

point

[in] O "peso" do ponto de 2/4-dígitos.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Symbol

Define o símbolo.

```
bool  Symbol(
   string    name     // symbol
   )
```

Parâmetros

name

[in] Símbolo (ativo).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A definição de trabalho do símbolo é necessário se o objeto utiliza o símbolo diferente do símbolo definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

Period

Define o timeframe.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // timeframe
   )
```

Parâmetros

value

[in] timeframe.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A definição de trabalho do timeframe é necessário se o objeto usa o timeframe diferente do que foi definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

Magic

Define o ID do Expert Advisor .

```
void  Magic(
   ulong  value     // magic
   )
```

Parâmetros

value

[in] ID do Expert Advisor.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SetPriceSeries

Define ponteiros para séries de preços externo.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // pointer
   CiHigh*    high,         // pointer
   CiLow*     low,          // pointer
   CiClose*   close         // pointer
   )
```

Parâmetros

open

[in] Ponteiro para timeseries Open.

high

[in] Ponteiro para timeseries High.

low

[in] Ponteiro para timeseries Low.

close

[in] Ponteiro para timeseries Close.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A configuração de ponteiros para timeseries externas (da série de preços) é necessária se o objeto usa timeseries do símbolo e timeframe diferente do símbolo e timeframe definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

SetOtherSeries

Define ponteiros para timeseries externos sem preço.

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // pointer
   CiTime*         time,          // pointer
   CiTickVolume*   tick_volume,   // pointer
   CiRealVolume*   real_volume    // pointer
   )
```

Parâmetros

spread

[in] Ponteiro para timeseries Spread.

time

[in] Ponteiro para timeseries Time.

tick_volume

[in] Ponteiro para timeseries TickVolume.

real_volume

[in] Ponteiro para timeseries RealVolume.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A configuração de ponteiros para timeseries externas (da série sem preço) é necessária se o objeto usa timeseries do símbolo e prazo diferente do símbolo e prazo definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Inicializa todos os indicadores e séries temporais.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries são inicializadas somente se o objeto usa o símbolo ou timeframe diferente do que foi definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

InitOpen

Inicialização das timeseries Open.

```
bool  InitOpen(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries Open são inicializadas apenas se o Expert Advisor usa o símbolo/timeframe, diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitHigh

Inicialização das timeseries High.

```
bool  InitHigh(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries High são inicializadas apenas se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitLow

Inicialização das timeseries Low .

```
bool  InitLow(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries Low são inicializadas apenas se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitClose

Inicialização das timeseries Close.

```
bool  InitClose(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries Close são inicializadas apenas se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitSpread

Inicialização das timeseries Spread.

```
bool  InitSpread(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries Spread são inicializadas somente se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitTime

Inicialização das timeseries Time.

```
bool  InitTime(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries Time são inicializadas somente se o Expert Advisor usa o símbolo/timeframe diferente do do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitTickVolume

Inicialização das timeseries TickVolume.

```
bool  InitTickVolume(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries TickVolume são inicializadas somente se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

InitRealVolume

Inicialização das timeseries RealVolume.

```
bool  InitRealVolume(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries RealVolume são inicializadas somente se o Expert Advisor usa o símbolo/timeframe diferente do que foi definido na inicialização (e se as timeseries estão sendo usadas ainda).

© 2000-2024, MetaQuotes Ltd.

PriceLevelUnit

Obtém a unidade de nível de preços.

```
virtual double  PriceLevelUnit()
```

Valor de retorno

O valor da unidade de Nível de Preços.

Observação

O método de uma classe base retorna o "peso" de 2/4 de pontos dos dígitos.

© 2000-2024, MetaQuotes Ltd.

StartIndex

Obtém o índice da barra para análise.

```
virtual int  StartIndex()
```

Valor de retorno

O índice de inicio da barra para analisar.

Observação

O método retorna 0 se o flag para analisar a barra atual é definido como verdadeiro (análise a partir da barra atual). Se o flag não é definido, retorna 1 (análise da última barra completa).

© 2000-2024, MetaQuotes Ltd.

CompareMagic

Compara a ID do Expert Advisor (mágico) com o valor determinado

```
virtual bool  CompareMagic(
   ulong  magic    // value to compare
   )
```

Parâmetros

magic

[in] Valor para comparar.

Valor de retorno

verdadeiro se eles forem iguais, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CExpert

CExpert é uma classe base para estratégias de negociação. Foi desenvolvida com algoritmos para trabalhar com séries temporais e indicadores e um conjunto de métodos virtuais para estratégia de negociação.

Como usar:

1. Preparar um algoritmo da estratégia;
2. Criar a própria classe, herdada da classe CExpert;
3. Substituir os métodos virtuais em sua classe com os seus próprios algoritmos.

Descrição

A classe CExpert é um conjunto de métodos virtuais para a implementação de estratégias de negociação.

Declaração

```
class CExpert : public CExpertBase
```

Título

```
#include <Expert\Expert.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpert

Métodos de classe

Inicialização	
Init	Instância de classe do método de inicialização.
virtual InitSignal	Inicializa o objeto Trading Signal
virtual InitTrailing	Inicializa o objeto Trailing Stop
virtual InitMoney	Inicializa o objeto Money Management
virtual InitTrade	Inicializa o objeto Trade
virtual ValidationSettings	Verifica as configurações
virtual InitIndicators	Inicializa indicadores e timeseries
virtual InitParameters	Método de inicialização dos parâmetros
virtual Deinit	Classe de instância do método de desinicialização
virtual DeinitSignal	Desinicializa o objeto Trading Signal

© 2000-2024, MetaQuotes Ltd.

Inicialização	
virtual DeinitTrailing	Desinicializa o objeto Trailing Stop
virtual DeinitMoney	Desinicializa o objeto Money Management
virtual DeinitTrade	Desinicializa o objeto Trade
virtual DeinitIndicators	Desinicializa indicadores e timeseries
Parâmetros	
Magic	Define a ID do Expert Advisor
MaxOrders	Obtém/Define a quantidade máxima de ordens permitidas
OnTickProcess	Define um flag para prosseguir o evento "OnTick"
OnTradeProcess	Define um flag para prosseguir o evento "OnTrade"
OnTimerProcess	Define um flag para prosseguir o evento "OnTimer"
OnChartEventProcess	Define um flag para prosseguir o evento "OnChartEvent"
OnBookEventProcess	Define um flag para prosseguir o evento "OnBookEvent"
Métodos de Processamento do Evento	
OnTick	Manipulador do evento OnTick
OnTrade	Manipulador do evento OnTrade
OnTimer	Manipulador do evento OnTimer
OnChartEvent	Manipulador do evento OnChartEvent
OnBookEvent	Manipulador do evento OnBookEvent
Métodos de atualização	
Refresh	Atualiza todos os dados
Processamento	
Processing	Algoritmo de processamento principal
Métodos de Entrada no Mercado	
CheckOpen	Verifica as condições de abertura da posição
CheckOpenLong	Verifica condições para abrir posição comprada
CheckOpenShort	Verifica condições para abrir posição vendida
OpenLong	Abre uma posição comprada
OpenShort	Abre uma posição vendida
Métodos de Saída do Mercado	
CheckClose	Verifica condições para fechar posição atual

© 2000-2024, MetaQuotes Ltd.

Inicialização	
CheckCloseLong	Verifica condições para fechar posição comprada
CheckCloseShort	Verifica condições para fechar posição vendida
CloseAll	Fecha a posição aberta e exclui todas as ordens
Close	Fecha a posição aberta
CloseLong	Fecha a posição comprada
CloseShort	Fecha a posição vendida
Métodos de Reversão de Posição	
CheckReverse	Verifica condições para reversão posição aberta
CheckReverseLong	Verifica condições para reversão posição comprada
CheckReverseShort	Verifica condições para reversão posição vendida
ReverseLong	Executa a operação de reversão da posição comprada
ReverseShort	Executa a operação de reversão da posição vendida
Posição/Ordem dos Métodos de Trailing	
CheckTrailingStop	Verifica as condições para modificar os parâmetros da posição
CheckTrailingStopLong	Verifica condições do Trailing Stop da posição comprada
CheckTrailingStopShort	Verifica condições do Trailing Stop da posição vendida
TrailingStopLong	Executa trailing stop para posição comprada
TrailingStopShort	Executa trailing stop para posição vendida
CheckTrailingOrderLong	Verifica condições do Trailing Stop das ordens pendentes Buy Limit/Stop
CheckTrailingOrderShort	Verifica condições do Trailing Stop das ordens pendentes Sell Limit/Stop
TrailingOrderLong	Executa Trailing Stop das ordens pendentes Buy Limit/Stop
TrailingOrderShort	Executa Trailing Stop das ordens pendentes Sell Limit/Stop
Método de Ordens Excluídas	
CheckDeleteOrderLong	Verifica condições para excluir uma ordem pendente Buy
CheckDeleteOrderShort	Verifica condições para excluir uma ordem pendente Sell
DeleteOrders	Exclui todas as ordens
DeleteOrder	Exclui ordem pendente Stop/Limit
DeleteOrderLong	Exclui ordem pendente Buy Limit/Stop

© 2000-2024, MetaQuotes Ltd.

Inicialização	
DeleteOrderShort	Exclui ordem pendente Sell Limit/Stop
Métodos de Volume de Negociação	
LotOpenLong	Obtém o volume de negociação para operação comprada
LotOpenShort	Obtém o volume de negociação para operação vendida
LotReverse	Obtém o volume de negociação para operação de reversão de posição
Métodos do Histórico da Negociação	
PrepareHistoryDate	Define data de início para o rastreamento do histórico de negociação
HistoryPoint	Cria um ponto de controle do histórico de negociação (salva número de posições, ordens, ofertas e ordens históricas)
CheckTradeState	Compara o estado atual com o que foi salvo e chama o manipulador de eventos correspondente
Flags de eventos	
WaitEvent	Define o flag de espera do evento
NoWaitEvent	Reseta o flag de espera do evento
Métodos de Processamento do Evento de Negociação	
TradeEventPositionStopTake	Manipulador do evento "Position Stop Loss/Take Profit triggered" (posições de Stop Loss ou Take Profit acionadas)
TradeEventOrderTriggered	Manipulador do evento "Pending Order Triggered" (ordem pendente acionada)
TradeEventPositionOpened	Manipulador do evento "Position Opened" (abertura de posição)
TradeEventPositionVolumeChanged	Manipulador do evento "Position Volume Changed" (volume da posição alterado)
TradeEventPositionModified	Manipulador do evento "Position Modified" (posição modificada)
TradeEventPositionClosed	Manipulador do evento "Position Closed" (posição fechada)
TradeEventOrderPlaced	Manipulador do evento "Pending Order Placed" (ordem pendente colocada)
TradeEventOrderModified	Manipulador do evento "Pending Order Modified" (ordem pendente modificada)

© 2000-2024, MetaQuotes Ltd.

Inicialização	
TradeEventOrderDeleted	Manipulador do evento "Pending Order Deleted" (ordem pendente excluída)
TradeEventNotIdentified	Manipulador do evento não identificado
Métodos de Serviço	
TimeframeAdd	Adiciona um timeframe para rastrear
TimeframesFlags	Obtém o flag indicando os timeframes com uma nova barra
SelectPosition	Seleção da posição para o subseqüente trabalho com ela.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

Init

Instância de classe do método de inicialização.

```
bool  Init(
   string              symbol,         // symbol
   ENUM_TIMEFRAMES     period,         // timeframe
   bool                every_tick,     // flag
   ulong               magic           // magic
   )
```

Parâmetros

symbol

[in] Símbolo (ativo).

period

[in] Timeframe (ENUM_TIMEFRAMES enumeration).

every_tick

[in] Flag.

magic

[in] ID doExpert Advisor (número mágico).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se every_tick é definido como verdadeiro, o método Processing() é chamado em cada tick do símbolo que está em atividade. Caso contrário, o Processing() é chamado apenas quando abre uma nova barra do símbolo que está em atividade.

© 2000-2024, MetaQuotes Ltd.

Magic

Define ID do Expert Advisor (mágico).

```
void  Magic(
   ulong  value     // new value
   )
```

Parâmetros

value

[in] O novo valor ID do Expert Advisor.

Valor de retorno

Nenhum.

Observação

Define o valor ID do Expert Advisor (mágico) para as seguintes classes: Trade, Signal, Money, Trailing.

Implementação

```
//+------------------------------------------------------------------+
//| Sets magic number for object and its dependent objects           |
//| INPUT:  value - new value of magic number.                       |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000-2024, MetaQuotes Ltd.

InitSignal

Inicializa objeto Trading Signal (sinal de negociação).

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // pointer
   )
```

Parâmetros

signal

[in] Ponteiro do objeto da classe CExpertSignal (ou seus herdeiros).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se o sinal for NULL, a classe CExpertSignal será usada e não faz nada.

© 2000-2024, MetaQuotes Ltd.

InitTrailing

Inicializa objeto Trailing Stop (arrastando o parar a perda) .

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // pointer
   )
```

Parâmetros

trailing

[in] Ponteiro do objeto da classe CExpertTrailing (ou seu herdeiro).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se o trailing for NULL, a classe ExpertTrailing será usada e não faz nada.

© 2000-2024, MetaQuotes Ltd.

InitMoney

Inicializa o objeto Money Management (gerenciamento do dinheiro).

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // pointer
   )
```

Parâmetros

money

[in] Ponteiro para o objeto da classe CExpertMoney (ou seu herdeiro).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se o dinheiro é NULL, a classe CExpertMoney vai ser usada e utiliza o lote mínimo.

© 2000-2024, MetaQuotes Ltd.

InitTrade

Inicializa o objeto Trade (negociação).

```
virtual bool  InitTrade(
   ulong            magic,        // magic
   CExpertTrade*    trade=NULL    // pointer
   )
```

Parâmetros

magic

[in] ID do Expert Advisor (vai ser usado em pedidos de negociação).

trade

[in] Ponteiro do objeto CExpertTrade.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Deinit

Classe de instância do método "deinitialization" (desinicialização).

```
virtual void Deinit()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnTickProcess

Define um flag para continuar o evento OnTick.

```
void  OnTickOProcess(
   bool     value        // flag
   )
```

Parâmetros

value

[in] Flag para continuar o evento OnTick.

Valor de retorno

Nenhum.

Observação

Se o flag é verdadeiro, o evento OnTick continua. Por padrão, o flag é definido como falso.

© 2000-2024, MetaQuotes Ltd.

OnTradeProcess

Define um flag para continuar o evento OnTrade.

```
void  OnTradeProcess(
   bool     value        // flag
   )
```

Parâmetros

value

[in] Flag para continuar o evento OnTrade.

Valor de retorno

Nenhum.

Observação

Se o flag é verdadeiro, o evento OnTrade continua. Por padrão, o flag é definido como falso.

© 2000-2024, MetaQuotes Ltd.

OnTimerProcess

Define um flag para continuar o evento OnTimer.

```
void  OnTimerProcess(
   bool     value        // flag
   )
```

Parâmetros

value

[in] Flag para continuar o evento OnTimer.

Valor de retorno

Nenhum.

Observação

Se o flag é verdadeiro, o evento OnTimer continua. Por padrão, o flag é definido como falso.

© 2000-2024, MetaQuotes Ltd.

OnChartEventProcess

Define um flag para continuar o evento OnChartEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Parâmetros

value

[in] Flag para continuar o evento OnChartEvent.

Valor de retorno

Nenhum.

Observação

Se o flag é verdadeiro, o evento OnChartEvent continua. Por padrão, o flag é definido como falso.

© 2000-2024, MetaQuotes Ltd.

OnBookEventProcess

Define um flag para continuar o evento OnBookEvent.

```
void  OnChartEventProcess(
   bool     value        // flag
   )
```

Parâmetros

value

[in] Flag para continuar o evento OnBookEvent.

Valor de retorno

Nenhum.

Observação

Se o flag é verdadeiro, o evento OnBookEvent continua. Por padrão o flag é definido como falso.

© 2000-2024, MetaQuotes Ltd.

MaxOrders (Método Get)

Obtém a quantidade máxima de ordens permitidas.

```
int  MaxOrders()
```

Valor de retorno

Quantidade máxima de ordens permitidas.

MaxOrders (Método Set)

Define a quantidade máxima de ordens permitidas.

```
void  MaxOrders(
   int     max_orders        // new value
   )
```

Parâmetros

max_orders

[in] O novo valor da quantidade máxima de ordens permitidas.

Valor de retorno

Nenhum.

Observação

Por padrão, a quantidade máxima de ordens permitidas = 1.

© 2000-2024, MetaQuotes Ltd.

Signal

Obtém o ponteiro do objeto Trade Signal (sinal de negociação).

```
CExpertSignal*  Signal() const
```

Valor de retorno

Ponteiro do objeto Trade Signal.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Também verifica as configurações de todos os objetos do Expert Advisor.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Inicia todos os indicadores e timeseries.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Os timeseries são inicializados se o objeto usa o símbolo ou o timeframe, diferente se símbolo ou timeframe é definido na inicialização. Conseqüentemente chama os métodos virtuais InitIndicators() dos objetos trading signal (sinal de negociação), trailing stop (arrastando a parada das perdas) e o money management (gestão do dinheiro).

© 2000-2024, MetaQuotes Ltd.

OnTick

Manipulador do evento OnTick.

```
virtual void  OnTick()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnTrade

Manipulador do evento OnTrade.

```
virtual void  OnTrade()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnTimer

Manipulador do evento OnTimer.

```
virtual void  OnTimer()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnChartEvent

Manipulador do evento OnChartEvent.

```
virtual void  OnChartEvent(
   const int     id,        // event id
   const long&   lparam,    // long parameter
   const double  dparam,    // double parameter
   const string  sparam     // string parameter
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long.

dparam

[in] parâmetro do evento tipo double.

sparam

[in] parâmetro do evento tipo string.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnBookEvent

Manipulador do evento OnBookEvent.

```
virtual void  OnBookEvent(
   const string&     symbol        // symbol
)
```

Parâmetros

symbol

[In] Símbolo do evento OnBookEvent.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

InitParameters

Inicia os parâmetros do Expert Advisor.

```
virtual bool  InitParameters()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A função InitParameters() da classe base CExpert não faz nada e sempre retorna verdadeira.

© 2000-2024, MetaQuotes Ltd.

DeinitTrade

Desinicializa o objeto Trade (negociação).

```
virtual void  DeinitTrade()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DeinitSignal

Desicinializa o objeto Signal.

```
virtual void  DeinitSignal()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DeinitTrailing

Desinicializa o objeto Trailing (arrastando).

```
virtual void  DeinitTrailing()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DeinitMoney

Desicinializa o objeto Money Management (gerenciamento de dinheiro).

```
virtual void  DeinitMoney()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DeinitIndicators

Desinicializa todos os indicadores e séries temporais.

```
virtual void  DeinitIndicators()
```

Valor de retorno

Nenhum.

Observação

Também desinicializa todos os indicadores e séries temporais de todos os objetos auxiliares.

© 2000-2024, MetaQuotes Ltd.

Refresh

Atualiza todos os dados.

```
virtual bool  Refresh()
```

Valor de retorno

verdadeiro se o processamento de tick é necessário, caso contrário será falso.

Observação

Permite determinar a necessidade do processamento de tick. Se necessário, atualiza todas as cotações, séries temporais, dados dos indicadores e retorna verdadeiro.

Implementação

```
//+------------------------------------------------------------------+
//| Refreshing data for processing                                   |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if successful, false otherwise.                     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- refresh rates
   if(!m_symbol.RefreshRates()) return(false);
//--- check need processing
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- refresh indicators
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000-2024, MetaQuotes Ltd.

Processing

Algoritmo de processamento principal.

```
virtual bool  Processing()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Ele faz as seguintes etapas:

1. Verificar a presença da posição aberta no símbolo. Se não abrir a posição, pule as etapas №2, №3 e №4.
2. Verificar condições para reversão da posição aberta (método CheckReverse()). Se a posição foi revertida, sair.
3. Condições para verificar o fechamento da posição (método CheckClose()). Se a posição foi fechada, pule a etapa №4.
4. Verificar as condições para modificar os parâmetros da posição (método CheckTrailingStop()). Se os parâmetros da posição foi modificado, sair.
5. Verificar a presença de ordens pendentes no símbolo. Se não houver qualquer ordem pendente, vá para a etapa №9.
6. Verificar condição para excluir a ordem (CheckDeleteOrderLong() para ordens pendentes de compra ou CheckDeleteOrderShort() para ordens pendentes de venda). Se a ordem foi excluída, vá para a etapa №9.
7. Verificar as condições para modificar parâmetros de ordem pendente (CheckTrailingOrderLong() para ordens de compra ou CheckTrailingOrderShort() para ordens de venda). Se os parâmetros de ordem foram modificados, sair.
8. Sair.
9. Verificar condições para abertura de posição (método CheckOpen()).

Se você quiser implementar seu próprio algoritmo, você precisará substituir o método da classe herdeira Processing() .

Implementação

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Main function                                                    |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if any trade operation processed, false otherwise.  |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- check if open positions
   if(m_position.Select(m_symbol.Name()))
     {
      //--- open position is available
      //--- check the possibility of reverse the position
      if(CheckReverse()) return(true);
      //--- check the possibility of closing the position/delete pending orders
      if(!CheckClose())
        {
         //--- check the possibility of modifying the position
         if(CheckTrailingStop()) return(true);
         //--- return without operations
         return(false);
        }
     }
//--- check if placed pending orders
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- check the ability to delete a pending order to buy
            if(CheckDeleteOrderLong()) return(true);
            //--- check the possibility of modifying a pending order to buy
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- check the ability to delete a pending order to sell
            if(CheckDeleteOrderShort()) return(true);
            //--- check the possibility of modifying a pending order to sell
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- return without operations
         return(false);
        }
     }
//--- check the possibility of opening a position/setting pending order
   if(CheckOpen()) return(true);
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

Seleção da posição para o subseqüente trabalho com ela.

```
void  SelectPosition()
```

valor de retorno

Nenhum.

Implementação

```
//+------------------------------------------------------------------+
//| Position select                                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpen

Verifica as condições para abertura da posição.

```
virtual bool  CheckOpen()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para abrir posições comprada (CheckOpenLong()) e vendida (CheckOpenShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for position open or limit/stop order set                  |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Verifica as condições para abrir posição comprada.

```
virtual bool  CheckOpenLong()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para abrir posição comprada (método CheckOpenLong() do objeto Signal) e satisfazendo a condição, abre a posição (método OpenLong()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for long position open or limit/stop order set             |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- check signal for long enter operations
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Verifica condições para abertura de posição vendida.

```
virtual bool  CheckOpenShort()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para abrir posição comprada (método CheckOpenShort() do objeto Signal) e satisfazendo a condição, abre a posição (método OpenShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for short position open or limit/stop order set            |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- check signal for short enter operations
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

OpenLong

Abre uma posição comprada.

```
virtual bool  OpenLong(
   double    price,     // price
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Parâmetros

price

[in] Price.

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Recebe o volume da negociação (método LotOpenLong(...)) e abre uma posição comprada (método Buy() do objeto Trade) se o volume da negociação não é igual a 0.

Implementação

```
//+------------------------------------------------------------------+
//| Long position open or limit/stop order set                       |
//| INPUT:  price - price,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- get lot for open
   double lot=LotOpenLong(price,sl);
//--- check lot for open
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

OpenShort

Abre uma posição vendida.

```
virtual bool  OpenShort(
   double    price,     // price
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Parâmetros

price

[in] Price.

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Recebe o volume da negociação (método LotOpenShort()) e abre uma posição vendida (chamando o método Sell do objeto Trade) se o volume da negociação não é igual a 0.

Implementação

```
//+------------------------------------------------------------------+
//| Short position open or limit/stop order set                      |
//| INPUT:  price - price,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- get lot for open
   double lot=LotOpenShort(price,sl);
//--- check lot for open
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverse

Verifica condições para reversão da posição aberta.

```
virtual bool  CheckReverse()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para reverter posições compradas (CheckReverseLong()) e vendidas (CheckReverseShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for position reverse                                       |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- check the possibility of reverse the long position
      if(CheckReverseLong())  return(true);
     }
   else
      //--- check the possibility of reverse the short position
      if(CheckReverseShort()) return(true);
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverseLong

Verifica condições para reversão de posição comprada.

```
virtual bool  CheckReverseLong()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para reversão de posição comprada (método CheckReverseLong()do objeto Signal) e satisfeita a condição, executa a operação de reversão da posição comprada atual (método ReverseLong (...)).

Implementação

```
//+------------------------------------------------------------------+
//| Check for long position reverse                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- check signal for long reverse operations
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckReverseShort

Verifica condições para reversão da posição vendida.

```
virtual bool  CheckReverseLong()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições para reversão de posição vendida (método CheckReverseShort()do objeto Signal) e satisfeita a condição, executa a operação de reversão da posição vendida atual (método ReverseShort()) se necessário.

Implementação

```
//+------------------------------------------------------------------+
//| Check for short position reverse                                 |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- check signal for short reverse operations
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

ReverseLong

Executa operação de reversão de posição comprada.

```
virtual bool  ReverseLong(
   double    price,     // price
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Parâmetros

price

[in] Preço.

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Obtém a reversão de posição do volume (método LotReverse()) e executa a operação de negociação de reversão da posição comprada (método Sell() do objeto Trade) se o volume de negociação não for igual a 0.

Implementação

```
//+-----------------------------------------------------------------+
//| Long position reverse                                           |
//| INPUT:  price - price,                                          |
//|         sl    - stop loss,                                      |
//|         tp    - take profit.                                    |
//| OUTPUT: true-if trade operation processed, false otherwise.     |
//| REMARK: no.                                                     |
//+-----------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- get lot for reverse
   double lot=LotReverse(sl);
//--- check lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

ReverseShort

Executa operação de reversão de posição vendida.

```
virtual bool  ReverseShort(
   double    price,     // price
   double    sl,        // Stop Loss
   double    tp         // Take Profit
   )
```

Parâmetros

price

[in] Price.

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Obtém a reversão de posição do volume (método LotReverse()) e executa a operação de negociação de reversão da posição vendida (método Buy() do objeto Trade) se o volume de negociação não for igual a 0.

Implementação

```
//+-----------------------------------------------------------------+
//| Short position reverse                                          |
//| INPUT:  price - price,                                          |
//|         sl    - stop loss,                                      |
//|         tp    - take profit.                                    |
//| OUTPUT: true-if trade operation processed, false otherwise.     |
//| REMARK: no.                                                     |
//+-----------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- get lot for reverse
   double lot=LotReverse(sl);
//--- check lot
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckClose

Verifica condições para fechar a posição atual.

```
virtual bool  CheckClose()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

1. Verifica condições do Stop Out do Expert Advisor (método CheckClose() do objeto de gestão de dinheiro). Se a condição for satisfeita, ele fecha as posições e exclui todas as ordens (CloseAll(...)).
2. Verifica as condições para fechar um posição comprada ou vendida (métodos CheckCloseLong () ou CheckCloseShort ()) e se a posição for fechada, exclui todas as ordens (método DeleteOrders()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for position close or limit/stop order delete              |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- position must be selected before call
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- check for position type
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- check the possibility of closing the long position / delete pending orders
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- check the possibility of closing the short position / delete pending order
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckCloseLong

Verifica condições para fechar a posição comprada.

```
virtual bool  CheckCloseLong()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica as condições para fechamento da posição comprada (método CheckCloseLong () do objeto Signal), satisfeita a condição, fecha a posição aberta (método CloseLong (...)).

Implementação

```
//+------------------------------------------------------------------+
//| Check for long position close or limit/stop order delete         |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- check for long close operations
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckCloseShort

Verifica condições para fechar a posição vendida.

```
virtual bool  CheckCloseShort()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica as condições para fechamento da posição vendida (método CheckCloseShort() do objeto Sinal), se satisfeita acondição, fecha a posição (método CloseShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for short position close or limit/stop order delete        |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- check for short close operations
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseAll

Executa parcialmente fechamento completo da posição.

```
virtual bool  CloseAll(
   double    lot    // lot
   )
```

Parâmetros

lot

[in] Número de lotes para reduzir a posição.

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Executa parcialmente fechamento completo da posição (métodos Sell() e Buy() do objeto da classe CTrade para as posições comprada/vendida) e exclui todas as ordens (método DeleteOrders ()).

Implementação

```
//+------------------------------------------------------------------+
//| Position close and orders delete                                 |
//| INPUT:  lot - volume for close.                                  |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- check for close operations
   if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);
   else                                             result=m_trade.Buy(lot,0,0,0);
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2024, MetaQuotes Ltd.

Close

Fecha a posição aberta.

```
virtual bool  Close()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Fecha a posição (método PositionClose() do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Position close                                                   |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseLong

Fecha a posição comprada.

```
virtual bool  CloseLong(
   double    price    // price
   )
```

Parâmetros

price

[in] Price.

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Fecha a posição comprada (método Sell(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Long position close                                              |
//| INPUT:  price - price for close.                                 |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

© 2000-2024, MetaQuotes Ltd.

CloseShort

Fecha a posição vendida.

```
virtual bool  CloseShort(
   double    price    // price
   )
```

Parâmetros

price

[in] Price.

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Fecha a posição vendida (método Buy (...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Short position close                                             |
//| INPUT:  price - price for close.                                 |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStop

Verifica as condições de Trailing Stop da posição aberta.

```
virtual bool  CheckTrailingStop()
```

Valor de retorno

verdadeiro se qualquer operação de negociação foi executada, caso contrário é falso.

Observação

Verifica as condições de Trailing Stop da posição aberta (CheckTrailingStopLong() ou CheckTrailingStopShort() para as posições compradas e vendidas).

Implementação

```
//+------------------------------------------------------------------+
//| Check for trailing stop/profit position                          |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- position must be selected before call
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- check the possibility of modifying the long position
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- check the possibility of modifying the short position
      if(CheckTrailingStopShort()) return(true);
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica as condições de Trailing Stop para abertura da posição comprada.

```
virtual bool  CheckTrailingStopLong()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica as condições de Trailing Stop para abertura da posição comprada (método CheckTrailingStopLong (...) do objeto Expert Trailing). Se as condições forem satisfeitas, e modifica os parâmetros da posição (método TrailingStopLong (...)).

Implementação

```
//+------------------------------------------------------------------+
//| Check for trailing stop/profit long position                     |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- check for long trailing stop operations
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- long trailing stop operations
      return(TrailingStopLong(sl,tp));
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica as condições de Trailing Stop para abertura da posição vendida.

```
virtual bool  CheckTrailingStopShort()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica as condições de Trailing Stop para abertura da posição vendida (método CheckTrailingStopShort (...) do objeto Expert Trailing). Se as condições forem satisfeitas, modifica os parâmetros da posição (método TrailingStopShort (...)).

Implementação

```
//+------------------------------------------------------------------+
//| Check for trailing stop/profit short position                    |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- check for short trailing stop operations
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- short trailing stop operations
      return(TrailingStopShort(sl,tp));
     }
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingStopLong

Modifica parâmetros de abertura da posição comprada.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parâmetros

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

A função modifica parâmetros de abertura da posição comprada (método PositionModify (...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Trailing stop/profit long position                               |
//| INPUT:  sl - new stop loss,                                      |
//|         tp - new take profit.                                    |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingStopShort

Modifica parâmetros abertura da posição vendida.

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parâmetros

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

tp

[in] Preço onde finaliza a posição com lucro (Take Profit).

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

A função modifica parâmetros de abertura da posição vendida (método PositionModify(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Trailing stop/profit short position                              |
//| INPUT:  sl - new stop loss,                                      |
//|         tp - new take profit.                                    |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderLong

Verifica condições do Trailing Stop das ordens pendentes Buy Limit/Stop.

```
virtual bool  CheckTrailingOrderLong()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica condições do Trailing Stop das ordens pendentes Buy Limit/Stop (método CheckTrailingOrderLong() do objeto Trade Signals) e se necessário modifica os parâmetros de ordem (método TrailingOrderLong (...)).

Implementação

```
//+------------------------------------------------------------------+
//| Check for trailing long limit/stop order                         |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- check the possibility of modifying the long order
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderShort

Verifica condições do Trailing Stop das ordens pendentes Sell Limit/Stop.

```
virtual bool  CheckTrailingOrderShort()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica condições do Trailing Stop das ordens pendentes Sell Limit/Stop (método CheckTrailingOrderShort() do objeto Trade Signals) e se necessário modifica os parâmetros de ordem (método TrailingOrderShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for trailing short limit/stop order                        |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- check the possibility of modifying the short order
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingOrderLong

Modifica parâmetros de ordem pendente Buy Limit/Stop.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Parâmetros

delta

[in] Preço delta.

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Modifica parâmetros de ordem pendente Buy Limit/Stop (método OrderModify(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Trailing long limit/stop order                                   |
//| INPUT:  delta - price change.                                    |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifying the long order
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2024, MetaQuotes Ltd.

TrailingOrderShort

Modifica parâmetros de ordem pendente bSell Limit/Stop.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Parâmetros

delta

[in] Preço delta.

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Modifica parâmetros de ordem pendente Sell Limit/Stop (método OrderModify(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Trailing short limit/stop order                                  |
//| INPUT:  delta - price change.                                    |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- modifying the short order
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckDeleteOrderLong

Verifica as condições para excluir a ordem pendente Buy Limit/Stop.

```
virtual bool  CheckDeleteOrderLong()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica o tempo de expiração da ordem. Verifica as condições para excluir a ordem pendente Buy Limit/Stop (método CheckCloseLong(...) da classe do objeto Signal). Exclui a ordem se a condição for satisfeita (método DeleteOrderLong()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for delete long limit/stop order                           |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- check the possibility of deleting the long order
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

CheckDeleteOrderShort

Verifica as condições para excluir a ordem pendente Sell Limit/Stop.

```
virtual bool  CheckDeleteOrderShort()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Verifica o tempo de expiração da ordem. Verifica as condições para excluir a ordem pendente Sell Limit/Stop (método CheckCloseShort(...) da classe do objeto Signal) e exclui a ordem se a condição for satisfeita (método DeleteOrderShort()).

Implementação

```
//+------------------------------------------------------------------+
//| Check for delete short limit/stop order                          |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation processed, false otherwise.      |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- check the possibility of deleting the short order
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- return without operations
   return(false);
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrders

Exclui todas as ordens.

```
virtual bool  DeleteOrders()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Exclui todas as ordens (DeleteOrder() para todos as ordens).

Implementação

```
//+------------------------------------------------------------------+
//| Delete all limit/stop orders                                     |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrder

Exclui a ordem pendente Limit/Stop.

```
virtual bool  DeleteOrder()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Exclui a ordem pendente Limit/Stop (método OrderDelete(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Delete limit/stop order                                          |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrderLong

Exclui a ordem pendente Buy Limit/Stop.

```
virtual bool  DeleteOrderLong()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Exclui a ordem pendente Buy Limit/Stop (método OrderDelete(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Delete long limit/stop order                                     |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

DeleteOrderShort

Exclui a ordem pendente Sell Limit/Stop.

```
virtual bool  DeleteOrderShort()
```

Valor de retorno

Verdadeiro se a operação de negociação foi executada, caso contrário falso.

Observação

Exclui a ordem pendente Sell Limit/Stop (método OrderDelete(...) do objeto da classe CTrade).

Implementação

```
//+------------------------------------------------------------------+
//| Delete short limit/stop order                                    |
//| INPUT:  no.                                                      |
//| OUTPUT: true-if trade operation successful, false otherwise.     |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotOpenLong

Obtém o volume de negociação para a operação de compra.

```
double  LotOpenLong(
   double    price,    // price
   double    sl        // Stop Loss
   )
```

Parâmetros

price

[in] Preço.

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

Valor de retorno

O volume da negociação (em lotes) para a operação comprada.

Observação

Recebe o volume da negociação para a operação de compra (método CheckOpenLong (...) do objeto de gestão de dinheiro).

Implementação

```
//+------------------------------------------------------------------+
//| Method of getting the lot for open long position.                |
//| INPUT:  price - price,                                           |
//|         sl    - stop loss.                                       |
//| OUTPUT: lot for open.                                            |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotOpenShort

Obtém o volume de negociação para a operação de venda.

```
double  LotOpenShort(
   double    price,    // price
   double    sl        // Stop Loss
   )
```

Parâmetros

price

[in] Preço.

sl

[in] Preço onde finaliza a perda (Stop Loss).

Valor de retorno

O volume da negociação (em lotes) para a operação de venda.

Observação

Recebe o volume da negociação para a operação de venda (método CheckOpenShort (...) do objeto da gestão do dinheiro).

Implementação

```
//+------------------------------------------------------------------+
//| Method of getting the lot for open short position.               |
//| INPUT:  price - price,                                           |
//|         sl    - stop loss.                                       |
//| OUTPUT: lot for open.                                            |
//| REMARK: no.                                                      |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

LotReverse

Obtém o volume de negociação para reversão de posição.

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Parâmetros

sl

[in] Preço onde finaliza a perda de uma posição (Stop Loss).

Valor de retorno

O (em lotes) .

Observação

Obtém o volume de negociação para operação de reversão de posição (método CheckReverse(...) do objeto da gestão de dinheiro).

Implementação

```
//+-----------------------------------------------------------------+
//| Method of getting the lot for reverse position.                 |
//| INPUT:  sl - stop loss.                                         |
//| OUTPUT: lot for open.                                           |
//| REMARK: no.                                                     |
//+-----------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2024, MetaQuotes Ltd.

PrepareHistoryDate

Configura a data inicial para rastreamento do histórico da negociação.

```
void  PrepareHistoryDate()
```

Observação

O período de acompanhamento do histórico da negociação é definido a partir do início do mês (mas não menos do que um dia).

© 2000-2024, MetaQuotes Ltd.

HistoryPoint

Cria um ponto de controle do histórico de negociação (salva número de posições, ordens, ofertas e ordens históricas).

```
void  HistoryPoint(
   bool   from_check_trade=false   // flag
   )
```

Parâmetros

from_check_trade=false

[in] Flag para evitar a recursividade.

Observação

Salva a quantidade de posições, ordens, ofertas e ordens históricas.

© 2000-2024, MetaQuotes Ltd.

CheckTradeState

Compara o estado atual com o que foi salvo e chama o manipulador de eventos correspondente.

```
bool CheckTradeState()
```

Valor de retorno

verdadeiro se o evento foi manipulado, caso contrário será falso.

Observação

Verifica o número de posições, ordens, ofertas e ordens históricas comparando com os valores, será salvo pelo método HistoryPoint (). Se a história de negociação foi alterada, chama o manipulador de eventos virtual correspondente.

© 2000-2024, MetaQuotes Ltd.

WaitEvent

Define o flag de espera do evento.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS     event          // flag
   )
```

Parâmetros

event

[in] Flag com eventos para definir (enumerador ENUM_TRADE_EVENTS).

Valor de retorno

Nenhum.

Flags de eventos

```
//--- flags of expected events
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // no expected events
   TRADE_EVENT_POSITION_OPEN         =0x1,       // flag of expecting the "opening of
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // flag of expecting of the "modifica
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // flag of expecting of the "modifica
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // flag of expecting of the "closing
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // flag of expecting of the "trigger
   TRADE_EVENT_ORDER_PLACE           =0x20,      // flag of expecting of the "placing
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // flag of expecting of the "modifica
   TRADE_EVENT_ORDER_DELETE          =0x80,      // flag of expecting of the "deletion
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // flag of expecting of the "trigger
  };
```

© 2000-2024, MetaQuotes Ltd.

NoWaitEvent

Reseta o flag da espera do evento.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS     event          // flag
   )
```

Parâmetros

event

[in] Flag com eventos para resetar (enumerador ENUM_TRADE_EVENTS).

Valor de retorno

Nenhum.

Flags de eventos

```
//--- flags of expected events
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT              =0,         // no expected events
   TRADE_EVENT_POSITION_OPEN         =0x1,       // flag of expecting the "opening of
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // flag of expecting of the "modifica
   TRADE_EVENT_POSITION_MODIFY       =0x4,       // flag of expecting of the "modifica
   TRADE_EVENT_POSITION_CLOSE        =0x8,       // flag of expecting of the "closing
   TRADE_EVENT_POSITION_STOP_TAKE    =0x10,      // flag of expecting of the "trigger
   TRADE_EVENT_ORDER_PLACE           =0x20,      // flag of expecting of the "placing
   TRADE_EVENT_ORDER_MODIFY          =0x40,      // flag of expecting of the "modifica
   TRADE_EVENT_ORDER_DELETE          =0x80,      // flag of expecting of the "deletion
   TRADE_EVENT_ORDER_TRIGGER         =0x100      // flag of expecting of the "trigger
  };
```

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionStopTake

Manipulador do evento "Position Stop Loss/Take Profit triggered" (posição Stop Loss/Take Profit alcançada) .

```
virtual bool  TradeEventPositionStopTake()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderTriggered

Manipulador do evento "Pending Order triggered" (ordem pendente alcançada).

```
virtual bool  TradeEventOrderTriggered()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionOpened

Manipulador do evento "Position opened" (posição aberta).

```
virtual bool  TradeEventPositionOpened()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

Manipulador do evento "Position volume changed" (volume da posição alterada).

```
virtual bool  TradeEventPositionVolumeChanged()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionModified

Manipulador do evento "Position modified" (posição modificada).

```
virtual bool  TradeEventPositionModified()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventPositionClosed

Manipulador do evento "Position closed (posição fechada).

```
virtual bool  TradeEventPositionClosed()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderPlaced

Manipulador do evento "Pending order placed" (ordem pendente colocada).

```
virtual bool  TradeEventOrderPlaced()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderModified

Manipulador do evento "Pending order modified" (ordem pendente modificada).

```
virtual bool  TradeEventOrderModified()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventOrderDeleted

Manipulador do evento "Pending order deleted" (exclusão de ordem pendente).

```
virtual bool  TradeEventOrderDeleted()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

TradeEventNotIdentified

Manipulador do evento não identificado.

```
virtual bool  TradeEventNotIdentified()
```

Valor de retorno

O método de classe CExpert não faz nada e sempre retorna verdadeiro.

Observação

Note que vários eventos de negociação podem chegar, nesse casos será difícil a identificáção.

© 2000-2024, MetaQuotes Ltd.

TimeframeAdd

Adiciona um timeframe para rastreamento.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // timeframe
   )
```

Parâmetros

period

[in] Timeframe (enumerador ENUM_TIMEFRAMES).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

TimeframesFlags

O método retorna o flag indicando os timeframes com uma nova barra.

```
int  TimeframesFlags(
   MqlDateTime&    time         // variable for time
   )
```

Parâmetros

time

[in] Variável do tipo MqlDateTime para novo tempo, transferido pela referência.

Valor de retorno

Retorna o flag que indica os timeframes com uma nova barra.

© 2000-2024, MetaQuotes Ltd.

CExpertSignal

CExpertSignal é uma classe base para sinais de negociação que não faz nada (exceto os métodos CheckReverseLong() e CheckReverseShort()), mas providencia as interfaces.

Como usar:

1. Preparar um algoritmo para sinais de negociação;
2. Criar a própria classe de sinal de negociação, herdado da classe CExpertSignal;
3. Substituir os métodos virtuais em sua classe com os seus próprios algoritmos.

Você pode encontrar exemplos de classes de sinal de negociação na pasta Expert\Signal\.

Descrição

CExpertSignal é uma classe base para implementação de algoritmos de sinal de negociação.

Declaração

```
class CExpertSignal : public CExpertBase
```

Título

```
#include <Expert\ExpertSignal.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertSignal

Descendentes diretos

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Métodos de classe

Inicialização	
virtual InitIndicators	Inicializa indicadores e séries temporais
virtual ValidationSettings	Verifica as configurações
virtual AddFilter	Adiciona um filtro ao sinal combinado
O acesso aos Dados Protegidos	
BasePrice	Define o nível de preço base
UsedSeries	Obtém os flags das timeseries usadas
Configuração de parâmetros	

© 2000-2024, MetaQuotes Ltd.

Inicialização	
Weight	Define o valor do parâmetro "Weight"
PatternsUsage	Define o valor do parâmetro "PatternsUsage"
General	Define o valor do parâmetro "General"
Ignore	Define o valor do parâmetro "Ignore"
Invert	Define o valor do parâmetro "Invert"
ThresholdOpen	Define o valor do parâmetro "ThresholdOpen"
ThresholdClose	Define o valor do parâmetro "ThresholdClose"
PriceLevel	Define o valor do parâmetro "pricelevel"
StopLevel	Define o valor do parâmetro "StopLevel"
TakeLevel	Define o valor do parâmetro "TakeLevel"
Expiration	Define o valor do parâmetro "Expiration"
Magic	Define o valor do parâmetro "Magic"
Verificando Condições de Negociação	
virtual CheckOpenLong	Verifica condições para abrir posição comprada
virtual CheckCloseLong	Verifica condições para fechar posição comprada
virtual CheckOpenShort	Verifica condições para abrir posição vendida
virtual CheckCloseShort	Verifica condições para fechar posição vendida
virtual CheckReverseLong	Verifica condições de reversão de posição comprada
virtual CheckReverseShort	Verifica condições de reversão de posição vendida
Configuração dos Parâmetros de Negociação	
virtual OpenLongParams	Define os parâmetros para abertura de posição comprada
virtual OpenShortParams	Define os parâmetros para abertura de posição vendida
virtual CloseLongParams	Define os parâmetros para o fechamento de posição comprada
virtual CloseShortParams	Define os parâmetros para o fechamento de posição vendida
Verificando Condições das Ordens Tipo Trailing	
virtual CheckTrailingOrderLong	Verifica condições para modificar os parâmetros de ordem pendente Buy
virtual CheckTrailingOrderShort	Verifica condições para modificar os parâmetros de ordem pendente Sell

© 2000-2024, MetaQuotes Ltd.

Inicialização	
Métodos de Verificar Formação de Ordens de Mercado	
virtual LongCondition	Obtém o resultado da verificação das condições de compra
virtual ShortCondition	Obtém o resultado da verificação das condições de venda
virtual Direction	Obtém a direção "ponderada" de preço

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

BasePrice

Define o nível de preço base.

```
void  BasePrice(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do nível de preço Base.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

UsedSeries

Obtém os flags das timeseries usadas.

```
int  UsedSeries()
```

Valor de retorno

Flags das timeserie usadas (se o símbolo/timeframe corresponde ao símbolo/timeframe que está trabalhando), caso contrário retorna 0 (zero).

© 2000-2024, MetaQuotes Ltd.

Weight

Define novo valor do parâmetro "Weight".

```
void  Weight(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor de peso.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

PatternUsage

Define o valor do parâmetro "PatternsUsage".

```
void  PatternUsage(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor dos padrões de uso.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

General

Define novo valor do parâmetro "General" .

```
void  General(
   int    value         // new value
   )
```

Parâmetros

value

[in] O novo valor "Geral".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Ignore

Define um novo valor de parâmetro "Ignore".

```
void  Ignore(
   long    value         // new value
   )
```

Parâmetros

value

[in] O novo valor "Ignorar".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Invert

Define novo valor do parâmetro "Invert".

```
void  Invert(
   long    value         // new value
   )
```

Parâmetros

value

[in] O novo valor para "Inverter".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

ThresholdOpen

Define novo valor do parâmetro "ThresholdOpen".

```
void  ThresholdOpen(
   long    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do limite para abertura.

Valor de retorno

Nenhum.

Observação

A variação do limite para abertura é de 0 a 100. Usado para abertura de posição.

© 2000-2024, MetaQuotes Ltd.

ThresholdClose

Define novo valor do parâmetro "ThresholdClose".

```
void  ThresholdOpen(
   long    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do limite para fechamento.

Valor de retorno

Nenhum.

Observação

A variação do limite para fechamento é de 0 a 100. Usado para fechar posição.

© 2000-2024, MetaQuotes Ltd.

PriceLevel

Define novo valor do parâmetro "pricelevel".

```
void  PriceLevel(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do nível de preço.

Valor de retorno

Nenhum.

Observação

O valor de "pricelevel" é definido em unidades de nível de preço. Os valores numéricos da unidade de nível de preços retorna pelo método PriceLevelUnit(). O "pricelevel" é usado para definir o preço de abertura em relação ao preço base.

© 2000-2024, MetaQuotes Ltd.

StopLevel

Define novo valor do parâmetro "StopLevel".

```
void  StopLevel(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do nível para finalizar a perda de uma negociação.

Valor de retorno

Nenhum.

Observação

O valor do "StopLevel" é definido em unidades de nível de preço. Os valores numéricos de unidade do nível de preço retorna pelo método PriceLevelUnit() . O "StopLevel" é usado para definir o nível de preço para finalizar a perda de uma negociação (Stop Loss) em relação ao preço de abertura da posição.

© 2000-2024, MetaQuotes Ltd.

TakeLevel

Define novo valor do parâmetro "TakeLevel".

```
void  TakeLevel(
   double    value         // new value
   )
```

Parâmetros

value

[in] O novo valor do nível para finalizar uma negociação com lucro.

Valor de retorno

Nenhum.

Observação

O valor de "TakeLevel" é definido em unidades de nível de preço. Os valores numéricos de unidade de nível de preços retorna pelo método PriceLevelUnit(). O "TakeLevel" é utilizado para definir o nível de preço para finalizar a negociação com lucro (Take Profit) em relação ao preço de abertura da posição.

© 2000-2024, MetaQuotes Ltd.

Expiration

Define o valor do parâmetro "Expiration" .

```
void  Expiration(
   int    value         // new value
   )
```

Parâmetros

value

[in] O novo valor da "Expiração".

Valor de retorno

Nenhum.

Observação

O valor do parâmetro "Expiration" é definido em barras. É usado como tempo de expiração para as Ordens Pendentes (quando de negociação com ordens pendentes).

© 2000-2024, MetaQuotes Ltd.

Magic

Define o valor do parâmetro "Magic".

```
void  Magic(
   int    value         // new value
   )
```

Parâmetros

value

[in] O novo valor de número mágico(ID do Expert Advisor).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Inicializa todos os indicadores e séries temporais.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para coleção de indicadores e timeseries.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

As timeseries são inicializadas somente se o objeto usa o símbolo ou timeframe diferente do que foi definido na inicialização.

© 2000-2024, MetaQuotes Ltd.

AddFilter

Adiciona um filtro ao sinal composto.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para filtrar objeto.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Verifica as condições para abrir posição comprada.

```
virtual bool  CheckOpenLong(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável do preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Verifica condições para abertura de posição vendida.

```
virtual bool  CheckOpenShort(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável do preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

OpenLongParams

Define os parâmetros para abertura de posição comprada.

```
virtual bool  OpenLongParams(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável do preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OpenShortParams

Define os parâmetros para abertura de posição vendida.

```
virtual bool  OpenShortParams(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável do preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CheckCloseLong

Verifica condições para fechar a posição comprada.

```
virtual bool  CheckCloseLong(
   double&  price      // price
   )
```

Parâmetros

price

[in][out] Variável para fechamento de preço, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CheckCloseShort

Verifica condições para fechar a posição vendida.

```
virtual bool  CheckCloseShort(
   double&  price      // price
   )
```

Parâmetros

price

[in][out] Variável para fechamento de preço, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CloseLongParams

Define os parâmetros para o fechamento de posição comprada.

```
virtual bool  CloseLongParams(
   double&    price          // price
   )
```

Parâmetros

price

[in][out] Variável para fechamento de preço, transmitida por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CloseShortParams

Define os parâmetros para o fechamento de posição vendida.

```
virtual bool  CloseShortParams(
   double&    price          // price
   )
```

Parâmetros

price

[in][out] Variável para fechamento de preço, transmitida por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CheckReverseLong

Verifica condições de reversão de posição comprada.

```
virtual bool  CheckReverseLong(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável do preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CheckReverseShort

Verifica condições de reversão de posição vendida.

```
virtual bool  CheckReverseShort(
   double&    price,          // price
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // expiration
   )
```

Parâmetros

price

[in][out] Variável da reversão de preço, transmitida por referência.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

expiration

[in][out] Variável para expiração do tempo, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderLong

Verifica condições para modificar os parâmetros de ordem pendente Buy.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // order
   double&         price           // price
   )
```

Parâmetros

order

[in] Ponteiro para objeto da classe COrderInfo.

price

[in][out] Variável de preço para Stop Loss.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingOrderShort

Verifica condições para modificar os parâmetros de ordem pendente.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // order
   double&         price           // price
   )
```

Parâmetros

order

[in] Ponteiro para objeto da classe COrderInfo.

price

[in][out] Variável de preço para Stop Loss.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

LongCondition

Verifica as condições para abrir posição comprada.

```
virtual int  LongCondition()
```

Valor de retorno

As condições são satisfeitas quando retorna o valor de 1 a 100 (dependendo da "força" do sinal). Retorna 0 (zero) quando não existe um sinal para abertura de posição comprada.

Observação

O método LongCondition() de uma classe base não tem nenhuma implementação a fim de verificar condições para abrir posição comprada e sempre retorna 0 (zero).

© 2000-2024, MetaQuotes Ltd.

ShortCondition

Verifica condições para abertura de posição vendida.

```
virtual int  ShortCondition()
```

Valor de retorno

As condições são satisfeitas quando retorna o valor de 1 a 100 (dependendo da "força" do sinal). Retorna 0 (zero) quando não existe um sinal para abertura de posição vendida.

Observação

O método ShortCondition() de uma classe base não tem nenhuma implementação a fim de verificar condições para abrir posição vendida e sempre retorna 0 (zero).

© 2000-2024, MetaQuotes Ltd.

Direction

Retorna o valor da direção "ponderada".

```
virtual double  Direction()
```

Valor de retorno

Ele retorna o valor>0 na direção ascendente (provavelmente) e retorna valor<0 na direção descendente. O valor absoluto depende da "força" de um sinal.

Observação

Se são usados os filtros, o resultado dependerá dos filtros.

© 2000-2024, MetaQuotes Ltd.

CExpertTrailing

CExpertTrailing é uma classe base para os algoritmos cujo os objetivos são mover o nível de preço que finaliza as perdas de uma negociação (Trailing Stop), esta classe não faz nada, mas providencia as interfaces.

Como usar:

1. Preparar um algoritmo para trailing;
2. Criar a sua própria classe de trailing, herdado da classe CExpertTrailing;
3. Substituir os métodos virtuais em sua classe com os seus próprios algoritmos.

Você pode encontrar exemplos da classe de trailing na pasta Expert\Trailing\.

Descrição

CExpertTrailing é uma classe base para implementação de algoritmos do Trailing Ttop.

Declaração

```
class CExpertTrailing : public CExpertBase
```

Título

```
#include <Expert\ExpertTrailing.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertTrailing

Descendentes diretos

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Métodos de classe

Verificação das Condições do Trailing Stop	
virtual CheckTrailingStopLong	Verifica condições para modificar os parâmetros da posição comprada
virtual CheckTrailingStopShort	Verifica condições para modificar os parâmetros da posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica condições para modificar os parâmetros da posição comprada.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // pointer
   double&         sl,            // Stop Loss
   double&         tp             // Take Profit
   )
```

Parâmetros

position

[in] Ponteiro para objeto da classe CPositionInfo.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

Observação

O método CheckTrailingStopLong (...) da classe base sempre retorna falso.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica condições para modificar os parâmetros da posição vendida.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // pointer
   double&         sl,            // Stop Loss
   double&         tp             // Take Profit
   )
```

Parâmetros

position

[in] Ponteiro para objeto da classe CPositionInfo.

sl

[in][out] Variável do preço de Stop Loss, transmitida por referência.

tp

[in][out] Variável do preço de Take Profit, transmitida por referência.

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

Observação

O método CheckTrailingStopShort(...) da classe base sempre retorna falso.

© 2000-2024, MetaQuotes Ltd.

CExpertMoney

CExpertMoney é uma classe base de algoritmos para dinheiro e gestão de risco.

Descrição

CExpertMoney é uma classe base para implementação de classe de dinheiro e gestão de risco.

Declaração

```
class CExpertMoney : public CObject
```

Título

```
#include <Expert\ExpertMoney.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney

Descendentes diretos

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Métodos de classe

O acesso aos Dados Protegidos	
Percent	Define o valor do parâmetro "Risk percent"
Inicialização	
virtual ValidationSettings	Verifica as configurações
Verificação das Condições de Negociação	
virtual CheckOpenLong	Obtém o volume para posição comprada
virtual CheckOpenShort	Obtém o volume para posição vendida
virtual CheckReverse	Obtém o volume para reversão de posição
virtual CheckClose	Verifica condições para fechar a posição aberta

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

Percent

Define o valor do parâmetro "Risk percent".

```
void  Percent(
   double  percent      // risk percent
   )
```

Parâmetros

percent

[in] Porcentagem de risco.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

O método ValidationSettings() da classe base sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // price
   double  sl          // Stop Loss
   )
```

Parâmetros

price

[in] preço de abertura da posição comprada.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume para a posição vendida.

```
virtual double  CheckOpenShort(
   double  price,      // price
   double  sl          // Stop Loss
   )
```

Parâmetros

price

[in] preço de abertura para a posição vendida.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para a posição vendida.

© 2000-2024, MetaQuotes Ltd.

CheckReverse

Obtém o volume para reversão de posição.

```
virtual double  CheckReverse(
   CPositionInfo*  position,      // pointer
   double          sl             // Stop Loss
   )
```

Parâmetros

position

[in] Ponteiro para objeto da classe CPositionInfo.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

Volume para reversão da posição.

© 2000-2024, MetaQuotes Ltd.

CheckClose

Verifica condições para fechar a posição aberta.

```
virtual double  CheckClose()
```

Valor de retorno

verdadeiro se a condição for satisfeita, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

Módulos de Sinais de Negociação

O terminal do cliente MT5 inclui um conjunto de módulos já prontos de sinais de negociação para o "Assistente MQL5". Ao criar um Expert Advisor no Assistente MQL5, você pode usar qualquer combinação de módulos de sinais de negociação (até 64). A decisão final sobre uma operação é feita com base na análise complexa dos sinais obtidos a partir de todos os módulos incluídos. Temos a descrição detalhada dos mecanismos das tomadas de decisões abaixo:

O terminal inclui os seguintes módulos de sinais:

- Os sinais do indicador Accelerator Oscillator
- Os sinais do indicador Adaptive Moving Average
- Os sinais do indicador Awesome Oscillator
- Os sinais do Oscillator Bears Power
- Os sinais do Oscillator Bulls Power
- Os sinais do Oscillator Commodity Channel Index
- Os sinais do oscilador DeMarker
- Os sinais do indicador Double Exponential Moving Average
- Os sinais do indicador Envelopes
- Os sinais do indicador Fractal Adaptive Moving Average
- Sinais do Filtro de Tempo Intradiário
- Os sinais do oscilador MACD
- Os sinais do indicador Moving Average
- Os sinais do indicador Parabolic SAR
- Os sinais do oscilador Relative Strength Index
- Os sinais do oscilador Relative Vigor Index
- Os sinais do oscilador Stochastic
- Os sinais do oscilador Triple Exponential Average
- Os sinais do indicador Triple Exponential Moving Average
- Os sinais do oscilador Williams Percent Range

Os Mecanismo das Tomadas de Decisões de Negociação com Base nos Módulos de Sinal

Os mecanismos das tomadas de decisões de negociação podem ser representados como a seguinte lista de princípios básicos:

- Cada um dos módulos de sinais tem o seu conjunto de módulos de mercado (certa combinação de preços e valores de um indicador).
- Cada modelo de mercado tem uma força determinada que pode variar de acordo com o intervalo de 1 a 100. Quanto maior é a força determinada, mais forte é o modelo.
- Cada um dos modelos gera uma previsão da direção do movimento dos preços.
- A previsão de um módulo é o resultado da pesquisa para os modelos incorporados, é transmitido como um número no intervalo de -100 a 100. O sinal determina a direção do movimento previsto

© 2000-2024, MetaQuotes Ltd.

(sinal negativo significa que o preço vai cair, sinal positivo significa que o preço vai subir). O valor absoluto corresponde à força do melhor modelo encontrado.

- A previsão de cada módulo é enviado para uma "decisão" final com um coeficiente de peso de 0-1 especificado em suas configurações ("Weight").
- O resultado da decisão é um número no intervalo de -100 a 100, onde o sinal determina o sentido do movimento previsto, o valor absoluto caracteriza a força do sinal. É calculado como uma média aritmética das previsões ponderadas de todos os módulos de sinais.

Cada Expert Advisor gerado tem duas configurações ajustáveis - níveis de limite de abertura e fechamento de uma posição (ThresholdOpen e ThresholdClose), que pode ser igual a um valor na faixa de 0 a 100. Se a intensidade de sinal final excede um nível limiar, uma operação de negociação que corresponde ao sinal será realizada.

Exemplos

Considere um Expert Advisor com os seguintes limiares: ThresholdOpen=20 e ThresholdClose=90. Dois módulos de sinais participam na tomada de decisões sobre as operações de negociação: o módulo MA com o peso de 0,4 e o módulo Stochastic com peso 0,8. Vamos analisar as duas variantes dos sinais obtidos:

Variante 1.

O preço cruzou para cima a MA em ascenção. Este caso corresponde a um dos modelos de mercado implementados no Módulo MA. Este modelo implica um aumento de preço. O seu significado é igual a 100. Ao mesmo tempo, o oscilador Stochastic está voltado para baixo e forma uma divergência com o preço. Este caso corresponde a um dos modelos implementados no Módulo Stochastic. Este modelo implica uma queda de preço. O peso deste modelo é 80.

Vamos calcular o resultado de "decisão" final. A taxa obtida a partir do módulo de MA é calculada como 0,4 * 100 = 40. O valor do módulo Stochastic é calculado como 0,8 * (-80) = -64. O valor final é calculado como a média aritmética das duas taxas: (40-64) / 2 = -12. O resultado da decisão é o sinal para a venda com força relativa igual a 12. O nível do limiar "ThresholdOpen" que é igual a 20 não é atingido. Deste modo a operação de negociação não será executada.

Variante 2.

O preço cruzou para baixo a MA em ascenção. Este caso corresponde a um dos modelos implementados no Módulo MA. Este Modelo implica um aumento de preço. O seu significado é igual a 10. Ao mesmo tempo, o oscilador Stochastic está voltado para baixo e forma uma divergência com o preço. Este caso corresponde a um dos modelos implementados no Módulo Stochastic. Este modelo implica uma queda de preço. O peso deste modelo é 80.

Vamos calcular o resultado de "decisão" final. A taxa obtidas a partir do módulo de MA é calculada como 0,4 * 10 = 4. O valor do módulo Stochastic é calculado como 0,8 * (-80) = -64. O valor final é calculado como uma média aritmética das duas taxas: (4-64) / 2 = -30. O resultado da decisão é o sinal para venda com força relativa igual a 30. O nível do limiar "ThresholdOpen" que é igual a 20 foi atingido. Assim, o resultado é um sinal para a abertura de uma posição vendida.

© 2000-2024, MetaQuotes Ltd.

a) Divergência do preço e o oscilador Stochastic (variantes 1 e 2).

b) O preço cruzou para cima a MA em ascenção (variante 1).

c) O preço cruzou para baixo a MA em ascenção(variantes 2).

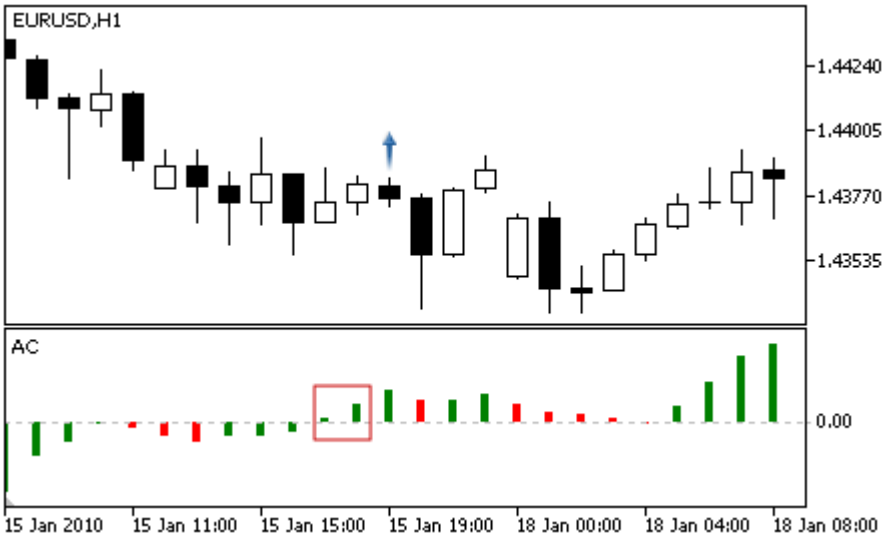
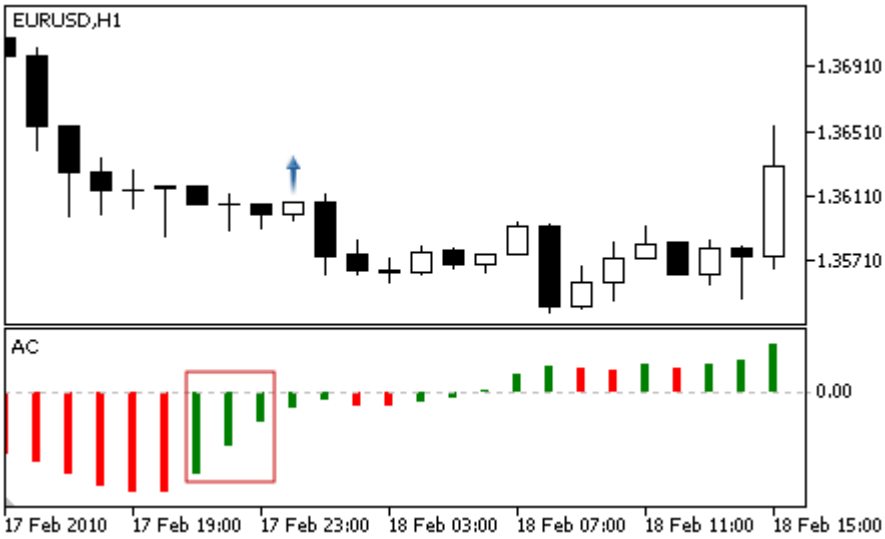
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Accelerator Oscillator

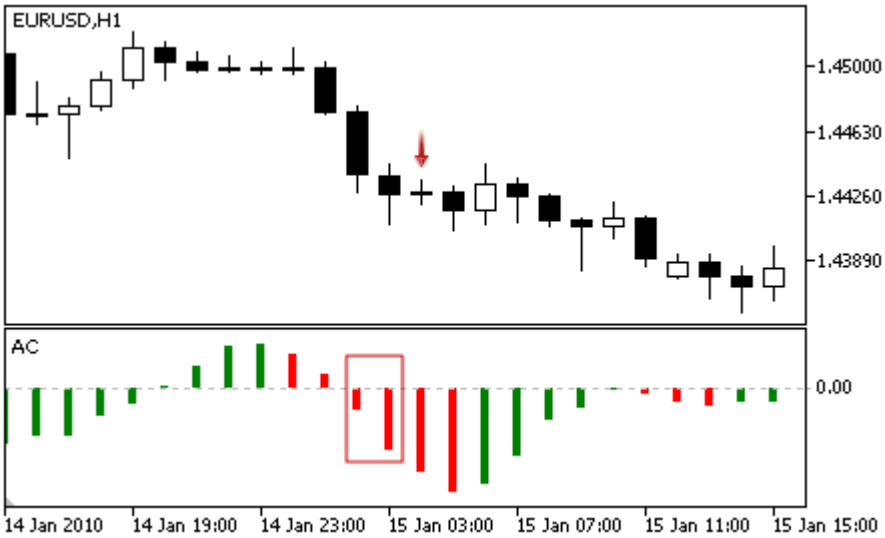
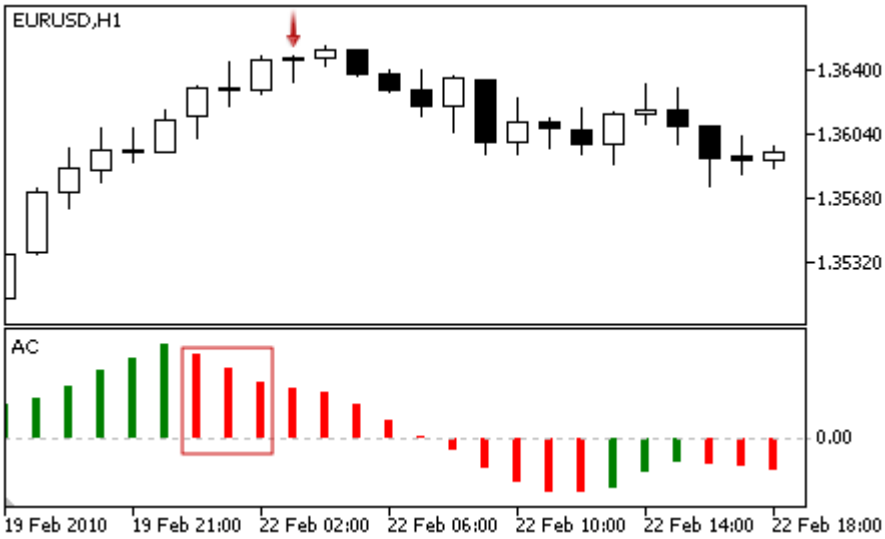
Este módulo é baseado nos modelos de mercado do indicador Accelerator oscilador. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O valor do indicador está acima de 0, se elevando nas barras analisadas e anteriores.  • O valor do indicador está abaixo de 0, se elevando nas barras analisadas e anteriores. 
Para Vender	• O valor do indicador é abaixo de 0, caindo nas barras analisadas e anteriores.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • O valor do indicador é acima de 0, caindo nas barras analisadas e anteriores 
Sem objeções para comprar	O valor do indicador cresce nas barras analisadas.
Sem objeções para vender	O valor do indicador decresce nas barras analisadas.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.

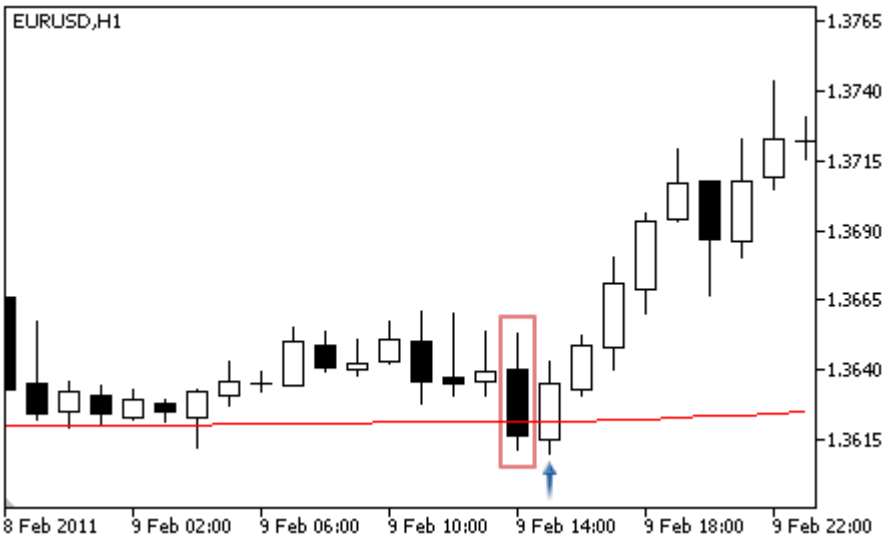
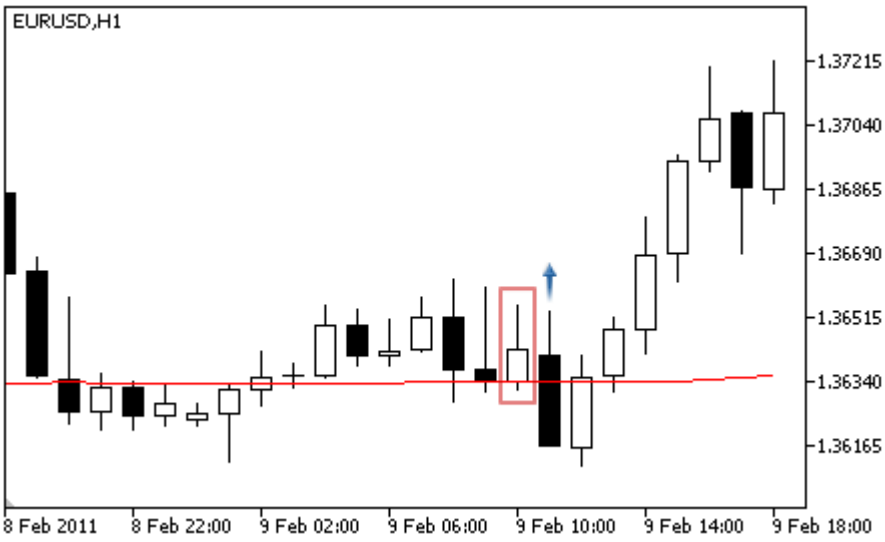
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Adaptive Moving Average

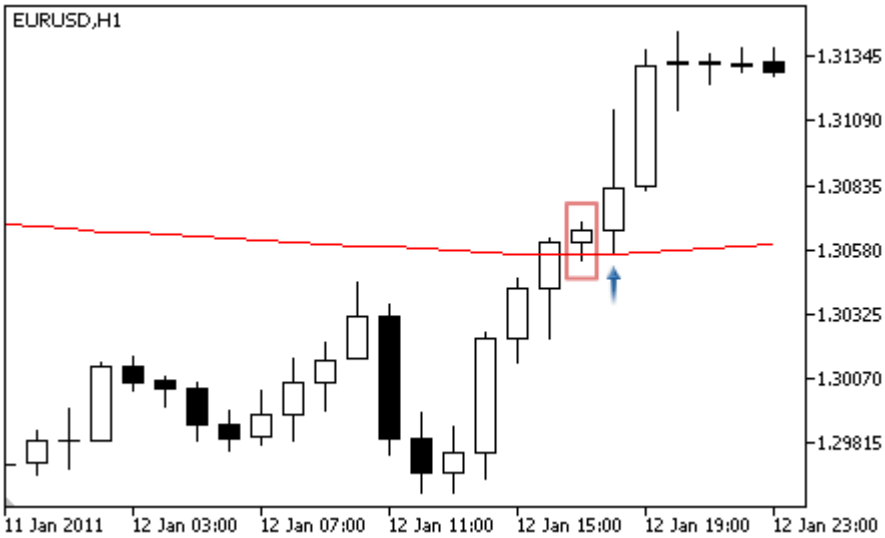
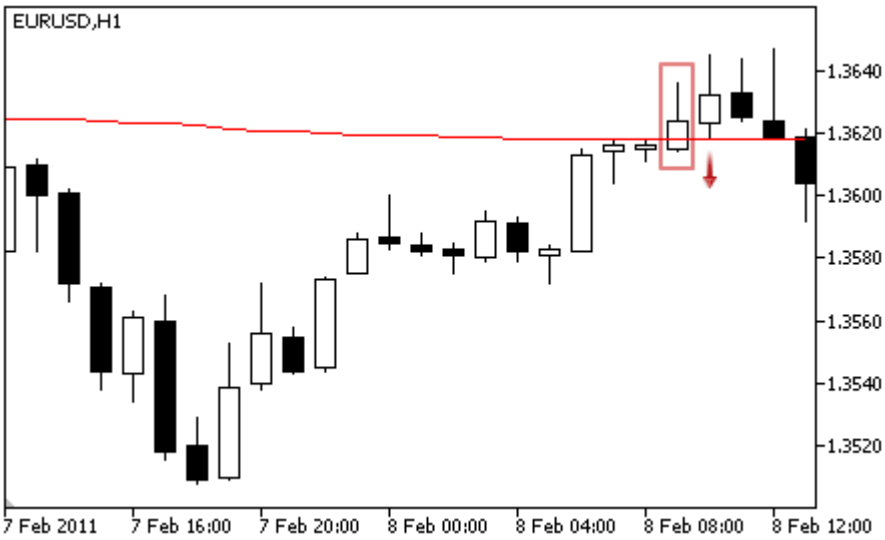
Este módulo é baseado nos modelos de mercado do indicador Adaptive Moving Average. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

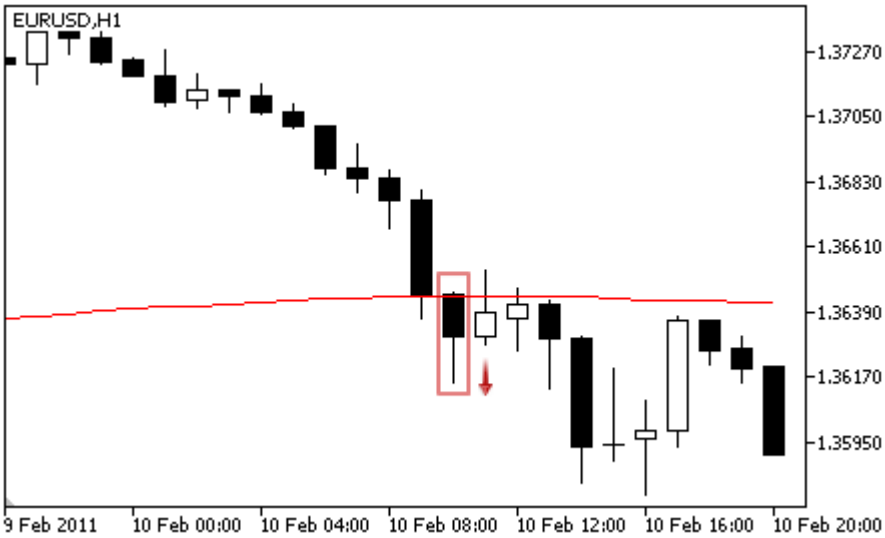
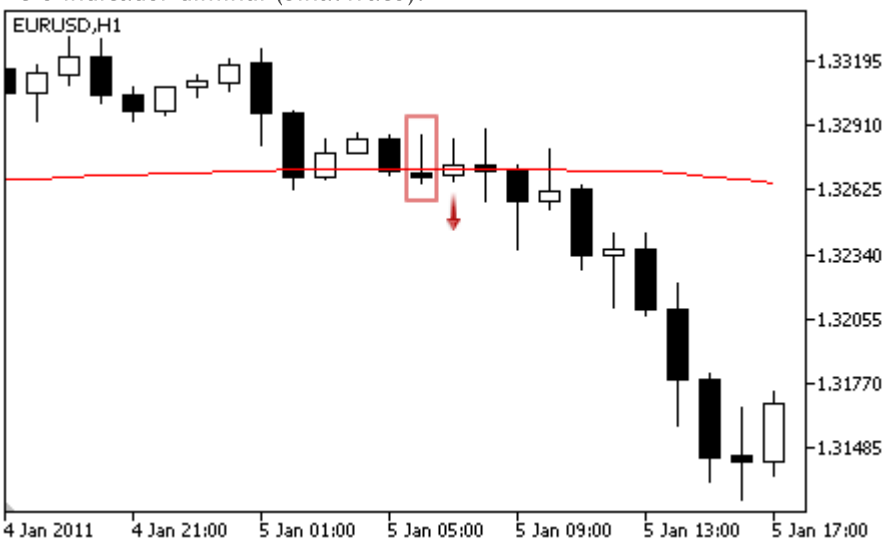
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para cima (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador aumenta (sinal forte). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• A sombra inferior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está acima do indicador, e a mínima do preço (Low) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco). 
Para Vender	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador diminui (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador diminui (sinal forte).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • A sombra superior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está abaixo do indicador, e a máxima do preço (High) está acima da linha do indicador) e o indicador diminui (sinal fraco). 
Sem objeções para comprar	Os preços estão acima do indicador.
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo da média do indicador.
Shift	Deslocamento do indicador ao longo do eixo do tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.

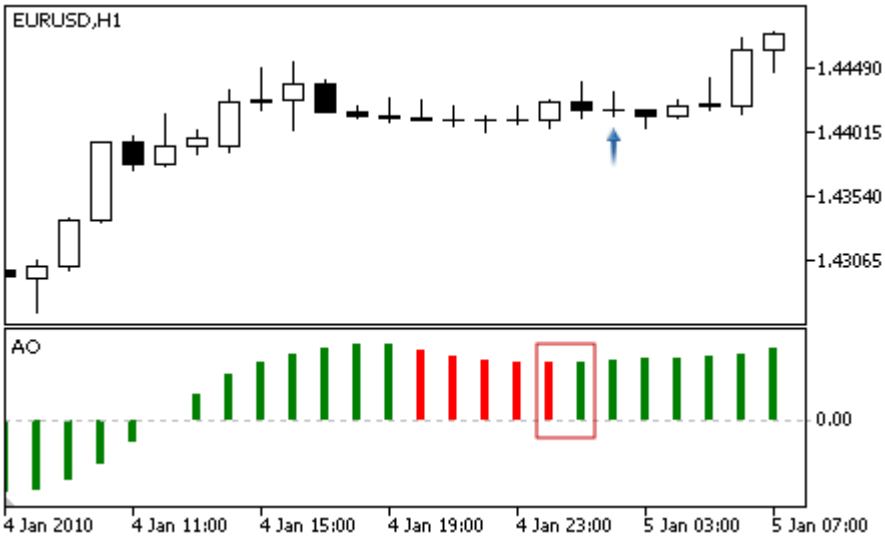
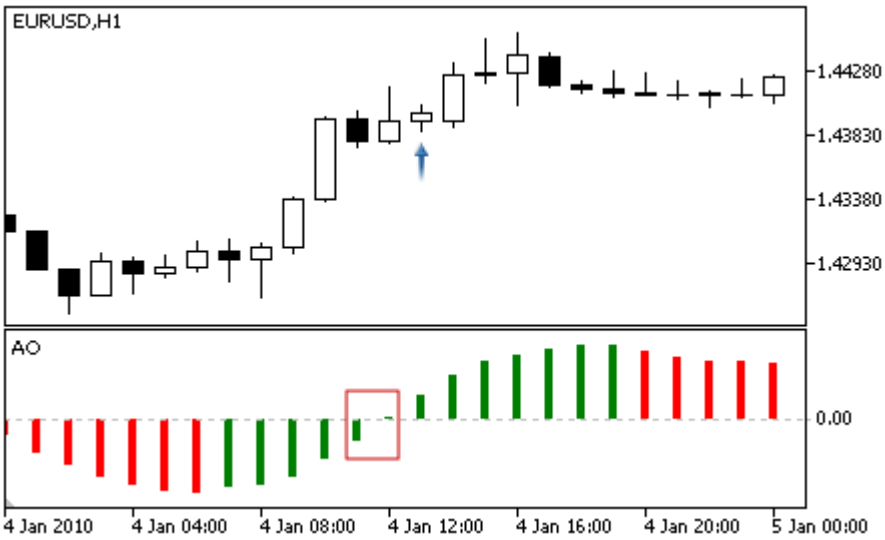
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Awesome Oscillator

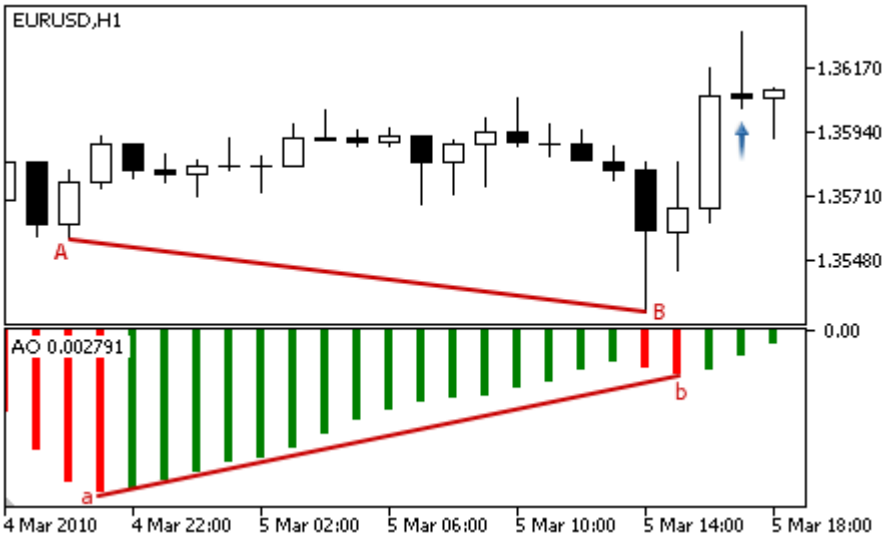
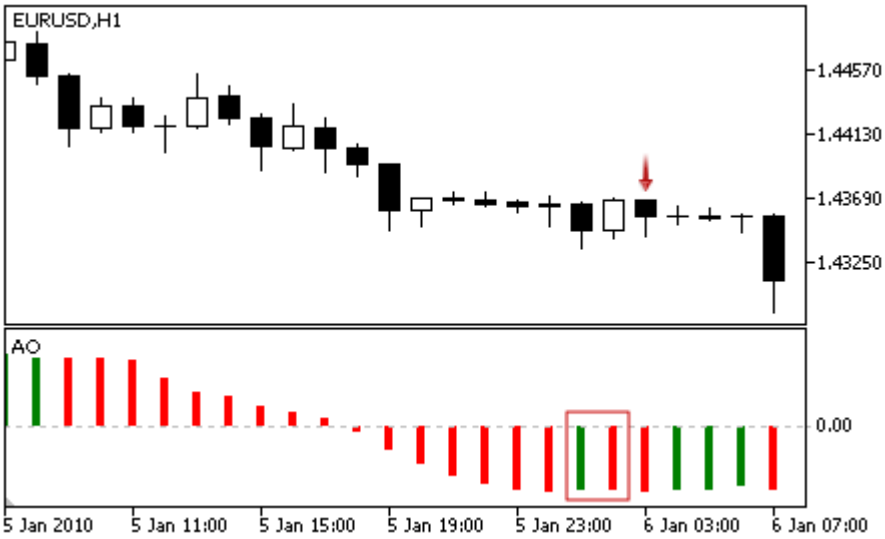
Este módulo de sinais baseia-se nos modelos de mercado do indicador Awesome Oscillator. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

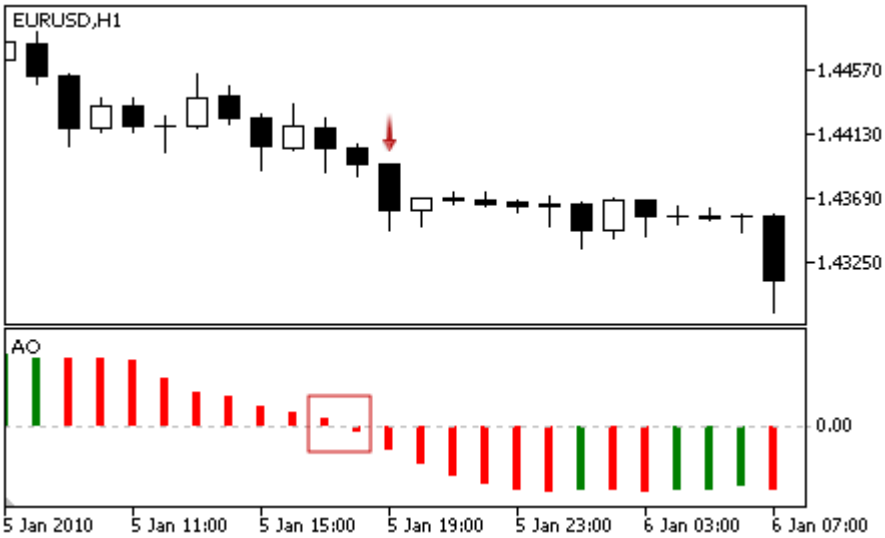
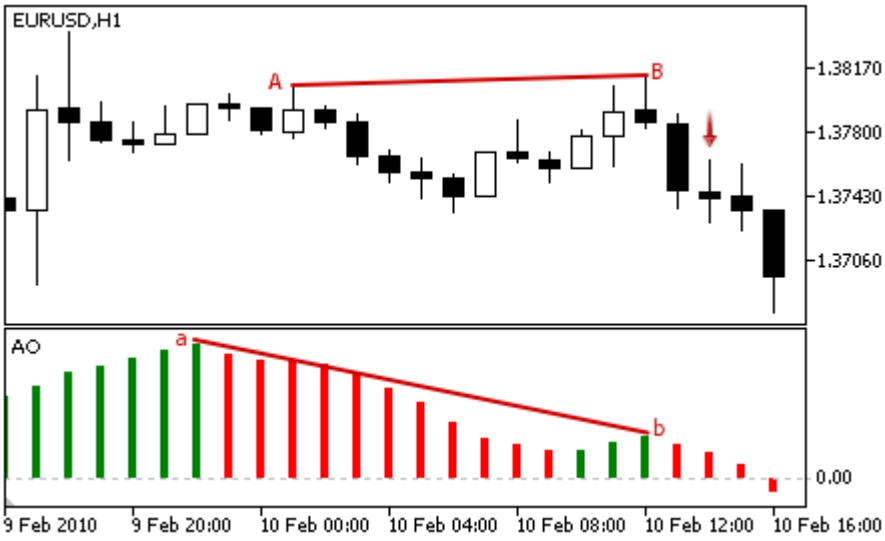
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Saucer** — Valor do indicador nas barra analisada aumenta e diminui nas barras anteriores; todos os valores estão acima de 0. (obs: Saucer seria a formação de um fundo tipo pires (prato/disco) nas extremidades das barras do indicador)  • **Cruzando a linha zero** — Valor do indicador é superior a 0 (zero) na barra analisada e menor do que 0 (zero) na barra anterior. 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• **Divergência** — o valor do fundo da primeira barra (a) analisada do indicador é menor do que o fundo da barra posterior (b) e o valor correspondente a mínima (Low) do preço da barra (A) é maior do que o valor correspondente a mínima (Low) do preço da barra (B). Além disso, as barras do indicador não devem estar acima do nivel zero. 
Para Vender	• **Saucer** — Valor do indicador nas barra analisada diminui e aumenta nas barras anteriores; todos os valores estão abaixo de 0. (obs: Saucer seria a formação de um fundo tipo pires (prato/disco) nas extremidades das barras do indicador)  • **Cruzando a linha zero** - Valor do indicador é inferior a 0 (zero) na barra analisada e maior do que 0 (zero) na barra anterior.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência** - o valor do topo da primeira barra (a) analisada do indicador é maior do que o topo da barra posterior (b) e o valor correspondente a máxima (High) do preço da barra (A) é menor do que o valor correspondente a máxima (High) do preço da barra (B). Além disso, as barras do indicador não devem estar abaixo do nível zero. 
Sem objeções para comprar	O valor do indicador cresce nas barras analisadas.
Sem objeções para vender	O valor do indicador decresce nas barras analisadas.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

© 2000-2024, MetaQuotes Ltd.

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.

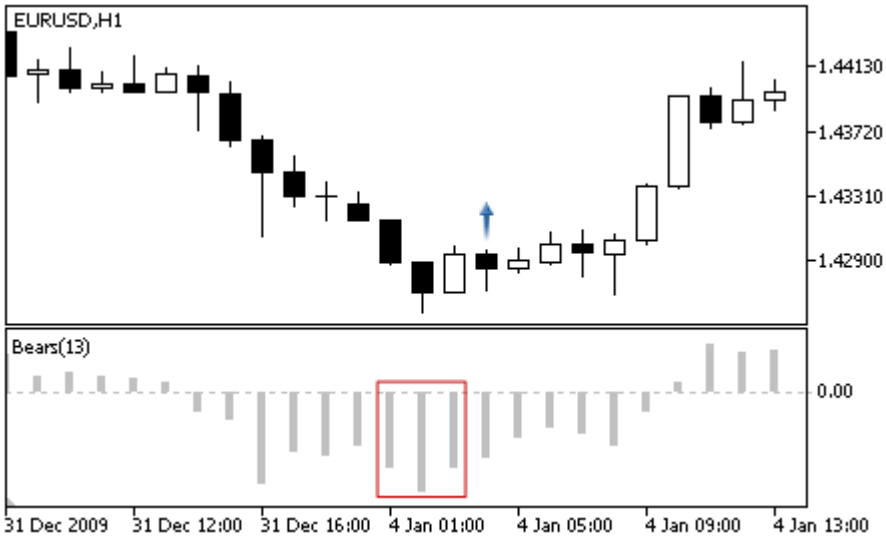
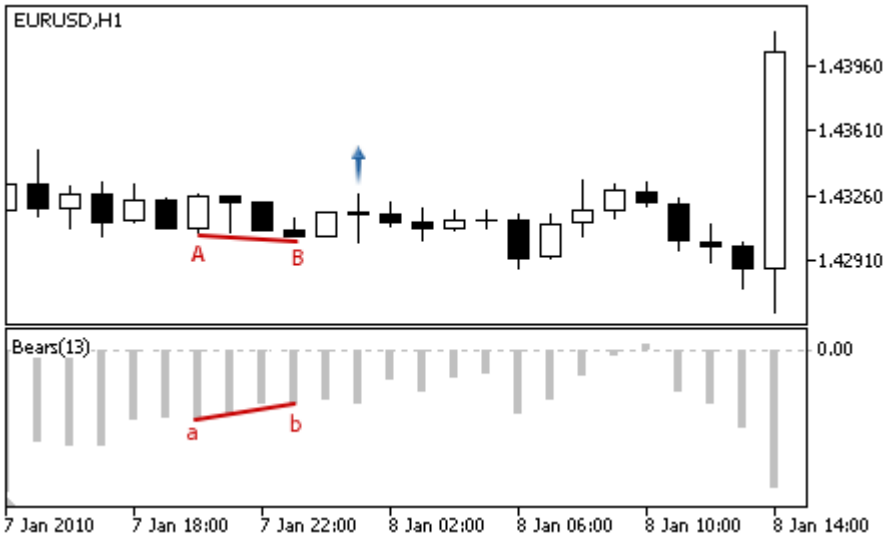
© 2000-2024, MetaQuotes Ltd.

Os sinais do Oscillator Bears Power

Este módulo de sinais baseia-se nos modelos de mercado do oscilador Bears Power. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão** - três barras do oscilador tem uma formação fractal tipo cunha e a barra analisada tem valor menor do que 0.  • **Divergência** - o valor do fundo da primeira barra (a) analisada do indicador é menor do que o fundo da barra posterior (b) e o valor correspondente a mínima (Low) do preço da barra (A) é maior do que o valor correspondente a mínima (Low) do preço da barra (B). Além disso, as barras do indicador não devem estar acima do nivel zero. 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
Para Vender	Não há sinais para venda.
Sem objeções para comprar	Valor do oscilador é menor do que 0.
Sem objeções para vender	Nenhum sinal.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodBears	Período de cálculo do oscilador.

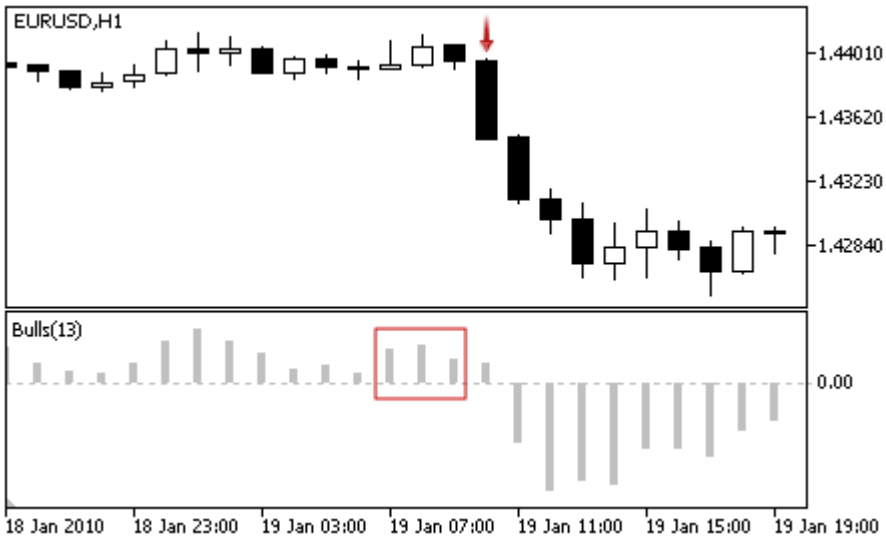
© 2000-2024, MetaQuotes Ltd.

Os sinais do Oscillator Bulls Power

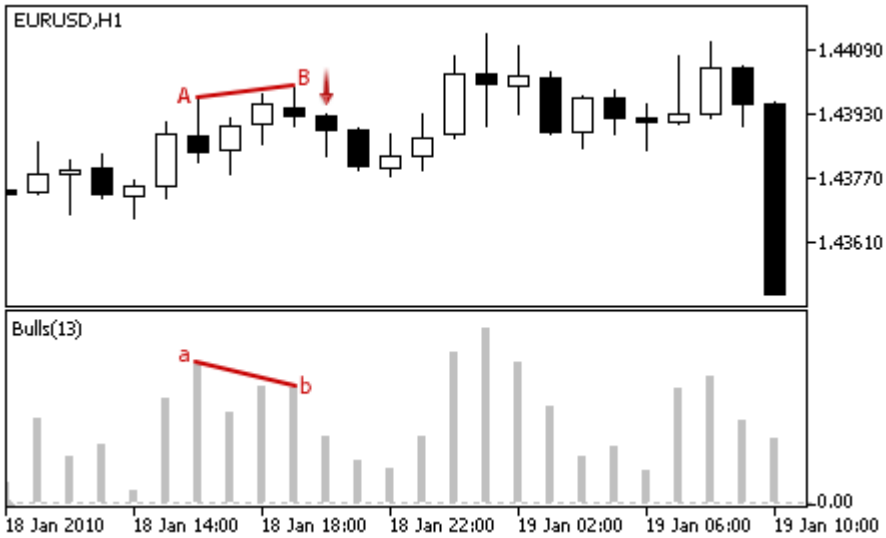
Este módulo de sinais baseia-se nos modelos de mercado do oscilador Bulls Power. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	Não há sinais de compra.
Para Vender	• **Reversão** - três barras do oscilador tem uma formação fractal tipo cunha e a barra analisada tem valor maior do que 0.  • **Divergência** - o valor do topo da primeira barra (a) analisada do indicador é maior do que o topo da barra posterior (b) e o valor correspondente a máxima (High) do preço da barra (A) é menor do que o valor correspondente a máxima (High) do preço da barra (B). Além disso, as barras do indicador não devem estar abaixo do nível zero. Além disso, o oscilador não deve descer abaixo do nível zero.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	
Sem objeções para comprar	Nenhum sinal.
Sem objeções para vender	Valor do oscilador é maior do que 0.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodBulls	Período de cálculo do oscilador.

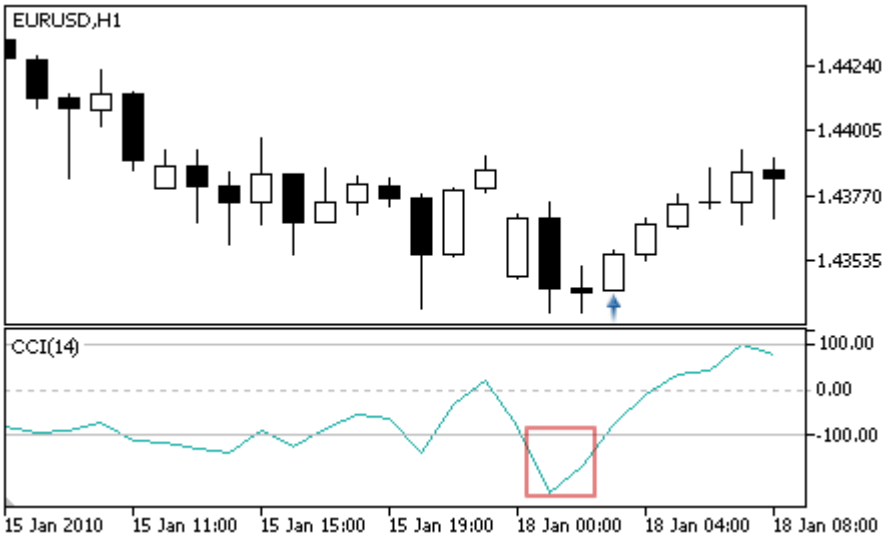
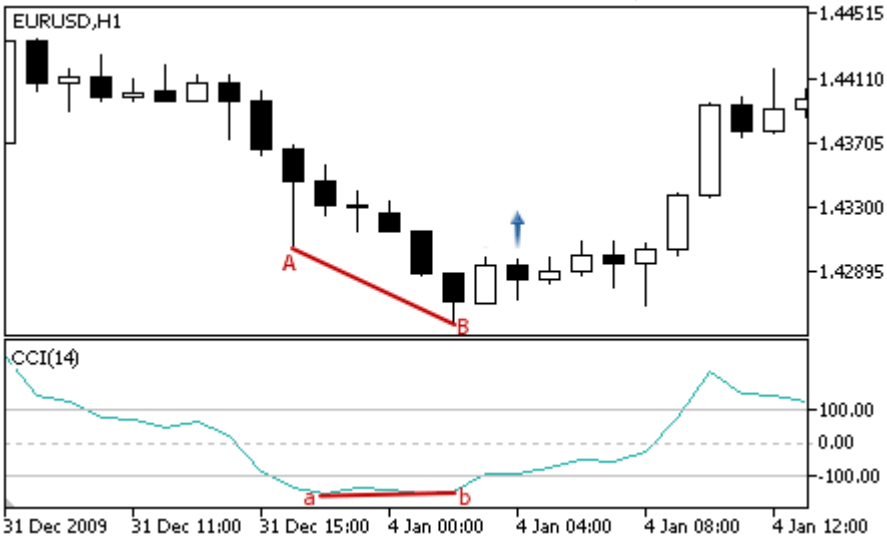
© 2000-2024, MetaQuotes Ltd.

Os sinais do Oscillator Commodity Channel Index

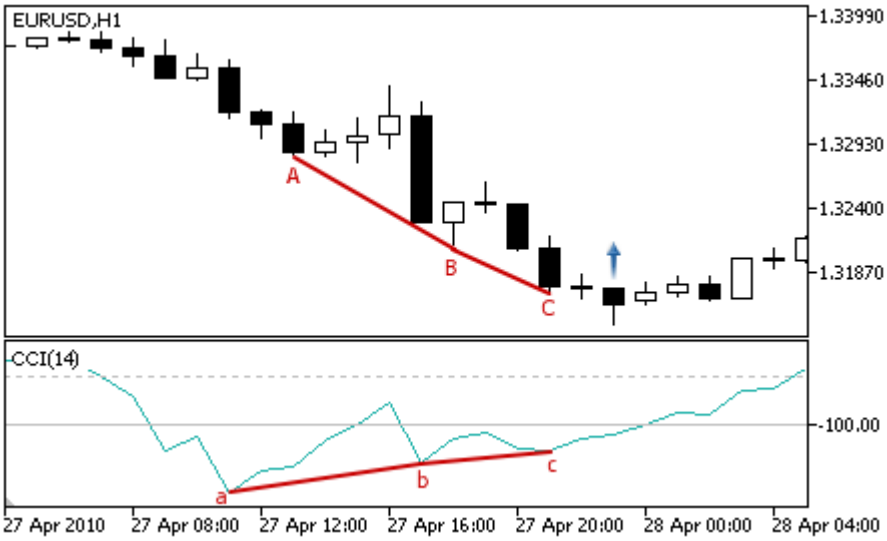
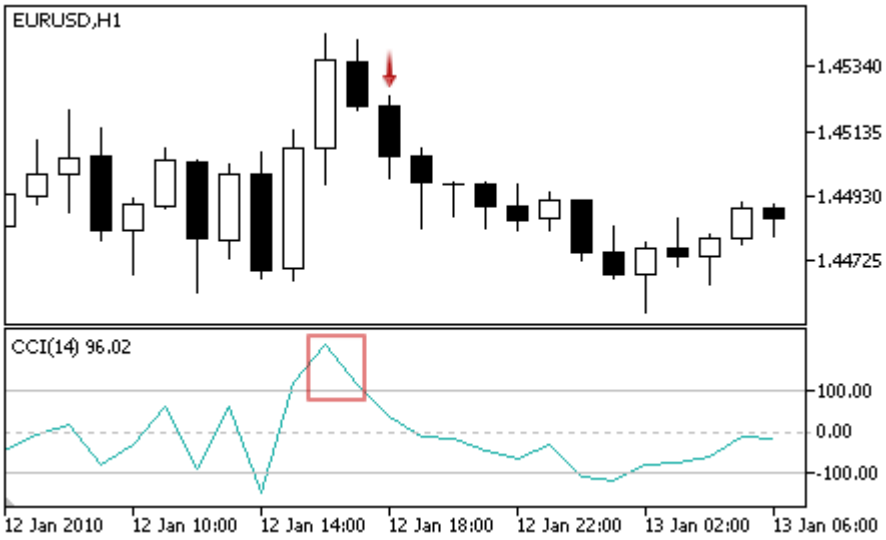
Este módulo de sinais baseia-se nos modelos de mercado do oscilador Commodity Channel Index. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

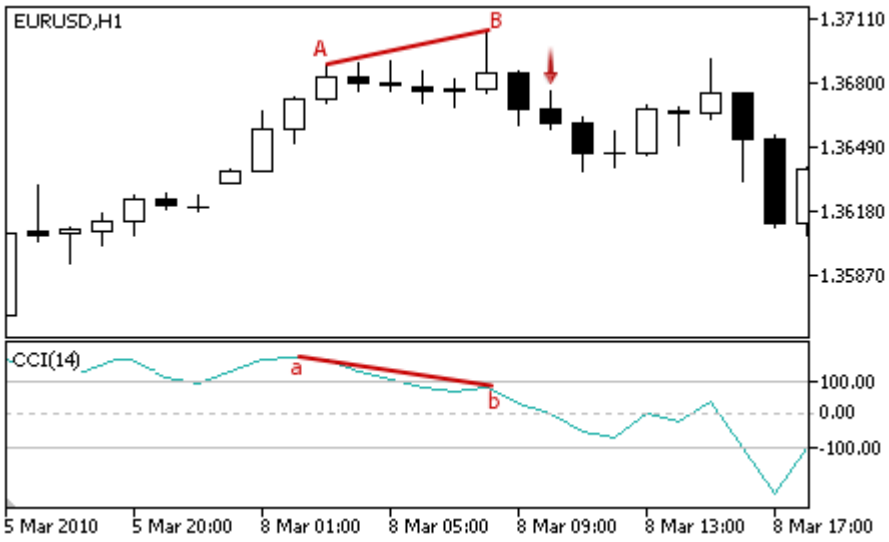
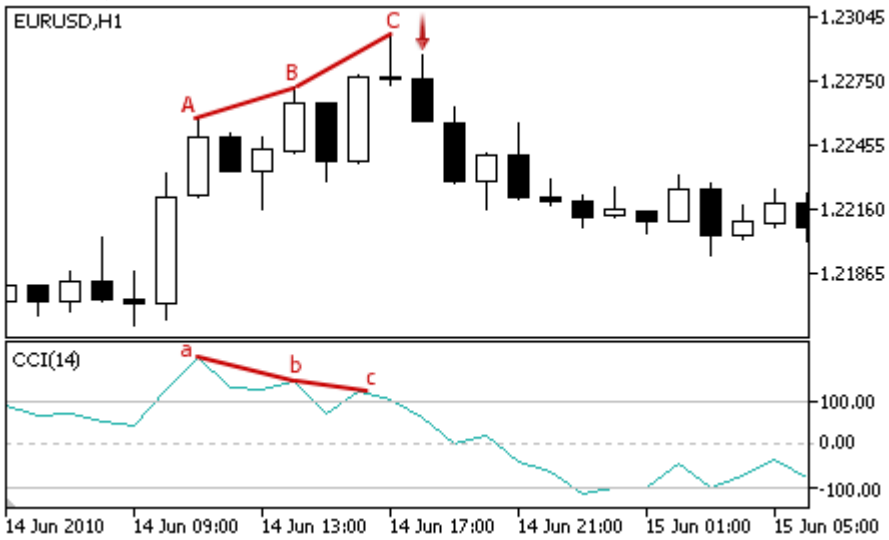
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão no nível sobrevendido** - O oscilador retorna para cima e seu valor na barra analisada está atrás do nível sobrevendido (valor padrão é -100).  • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• **Divergência Dupla** - a linha do oscilador forma três consequentes fundos (a/b/c), onde o valor de a < b < c; correspondendo a três mínimas (Low) de preço das barras formadas (A/B/C), onde o valor da mínima do preço A > B > C. 
Para Vender	• **Reversão no nível sobrecomprado** - O oscilador retorna para baixo e seu valor na barra analisada está atrás do nível sobrecomprado (valor padrão é 100).  • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência Dupla** - a linha do oscilador forma três consequentes topos (a/b/c), onde o valor de a > b > c; correspondendo a três máximas (High) de preço das barras formadas (A/B/C), onde o valor da máxima do preço A < B < C. 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodCCI	Período de cálculo do oscilador.
Applied	As séries de preços utilizadas para o cálculo do oscilador.

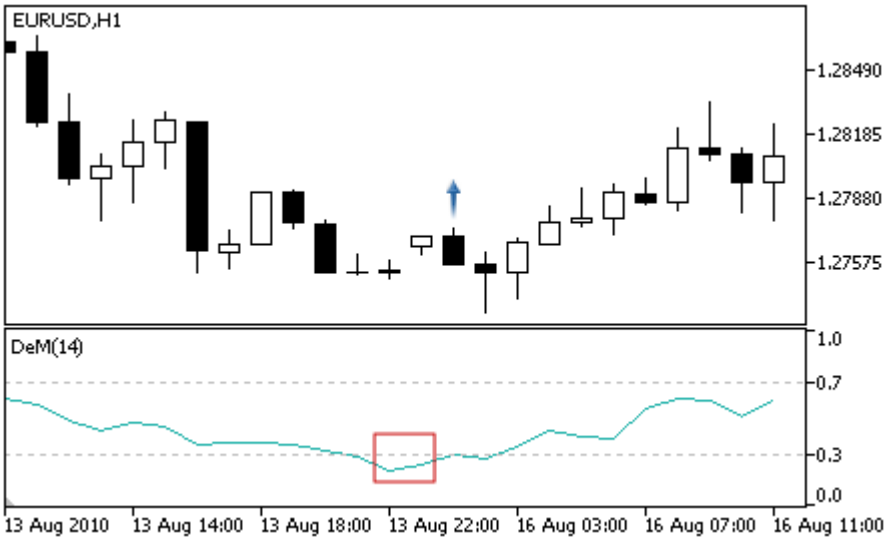
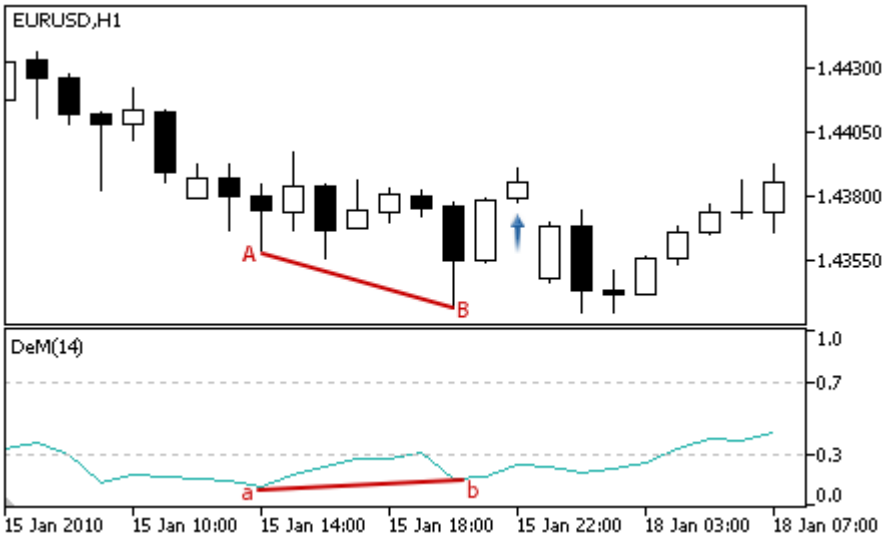
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador DeMarker

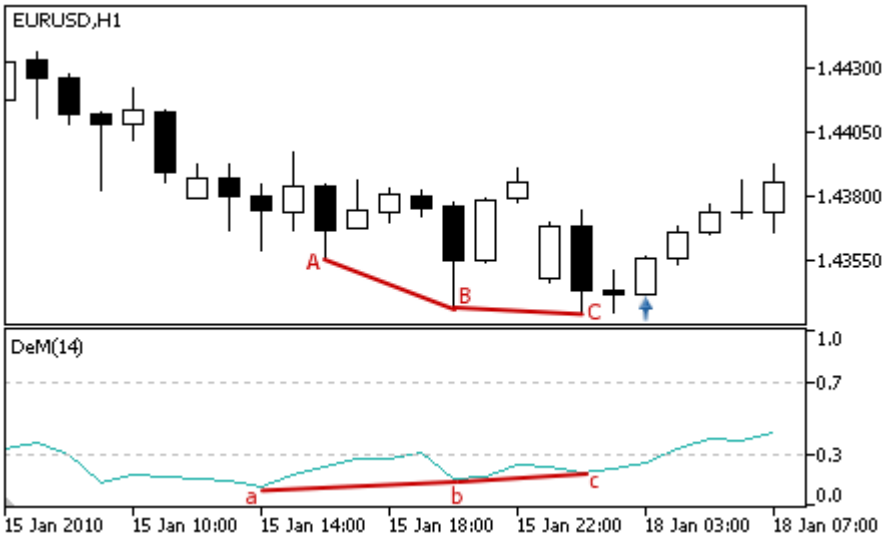
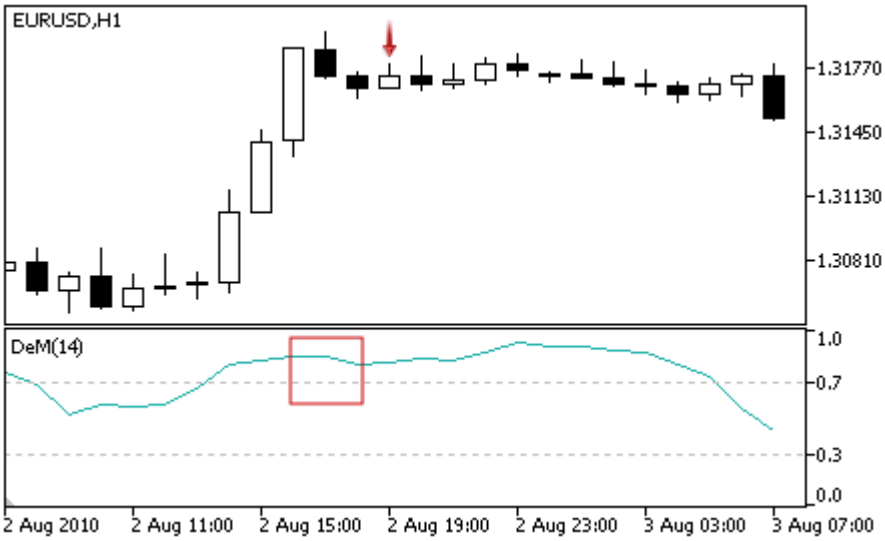
Este módulo de sinais baseia-se nos modelos de mercado do oscilador DeMarker. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

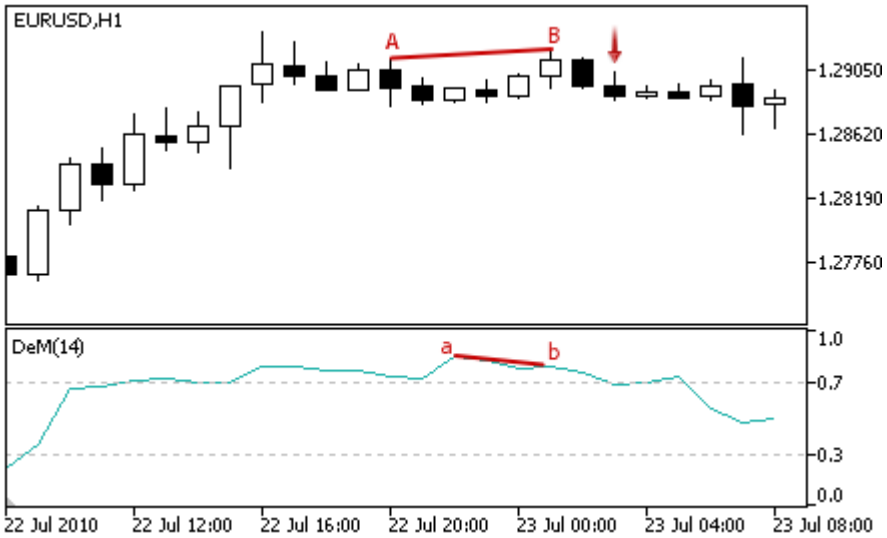
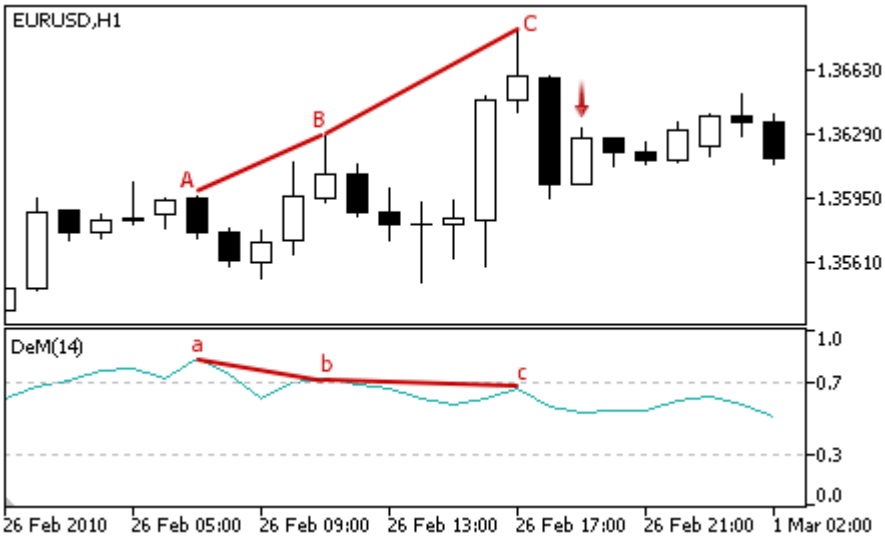
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão atrás do nível sobrevendido** - O oscilador se volta para cima e seu valor na barra analisada está atrás do nível de sobrevendido (valor padrão é 0,3).  • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• **Divergência Dupla** - a linha do oscilador forma três consequentes fundos (a/b/c), onde o valor de a < b < c; correspondendo a três mínimas (Low) de preço das barras formadas (A/B/C), onde o valor da mínima do preço A > B > C. 
Para Vender	• **Reversão atrás do nível sobrecomprado** - O oscilador se volta para baixo e seu valor na barra analisada está atrás do nível sobrecomprado (valor padrão é 0,7).  • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência Dupla** - a linha do oscilador forma três consequentes topos (a/b/c), onde o valor de a > b > c; correspondendo a três máximas (High) de preço das barras formadas (A/B/C), onde o valor da máxima do preço A < B < C. 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodDeM	Período de cálculo do oscilador.

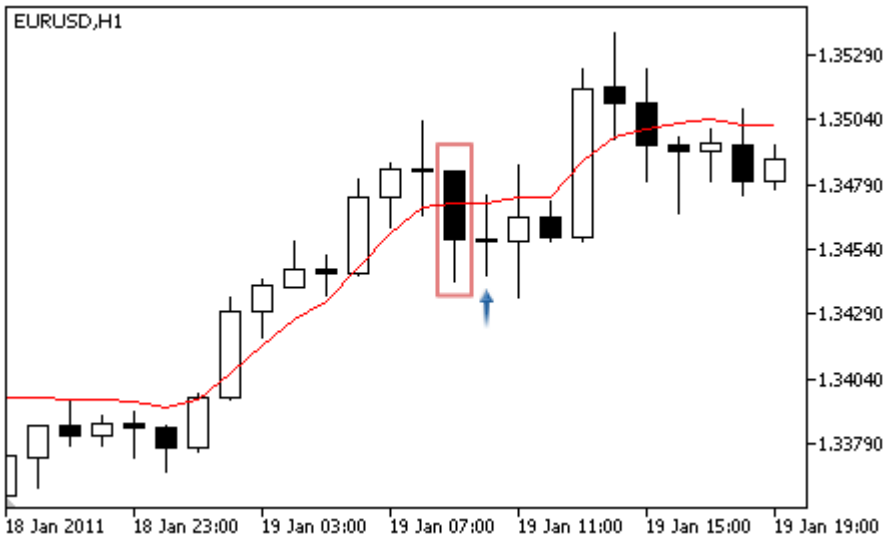
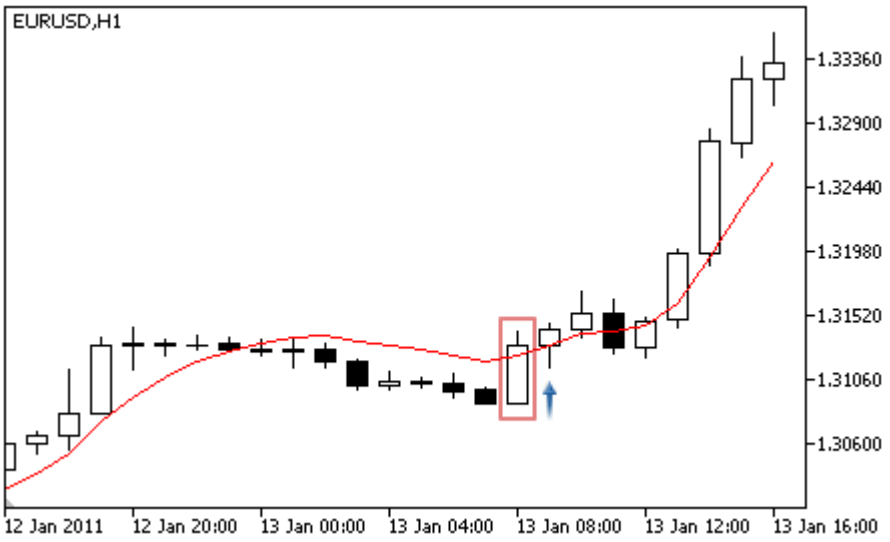
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Double Exponential Moving Average

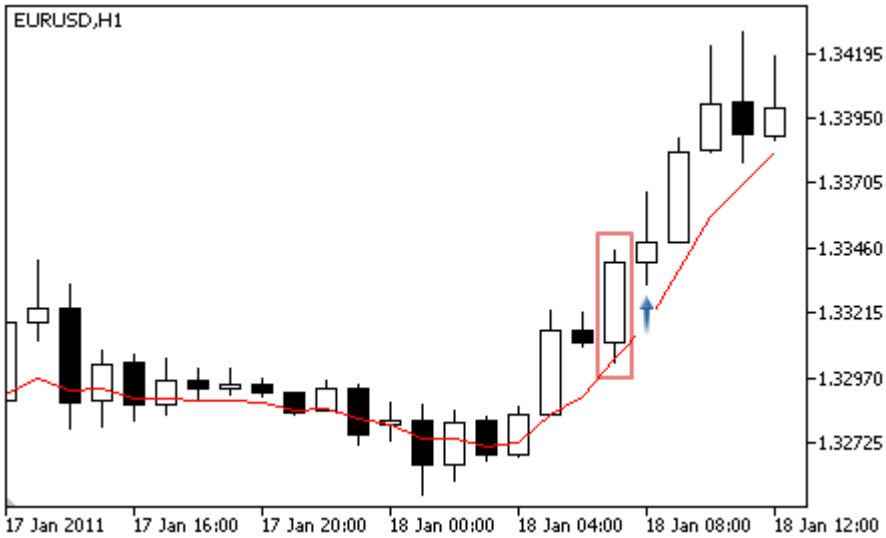
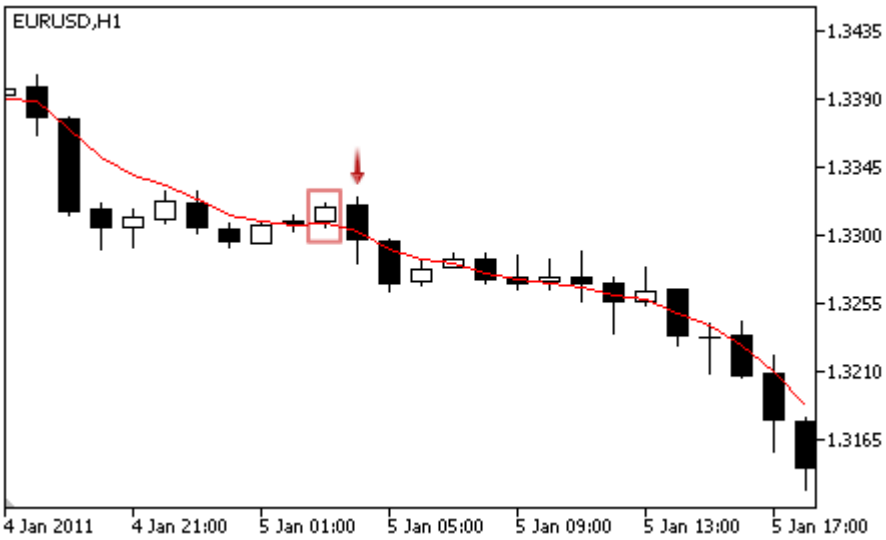
Este módulo é baseado nos modelos de mercado do indicador Double Exponential Moving Average. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

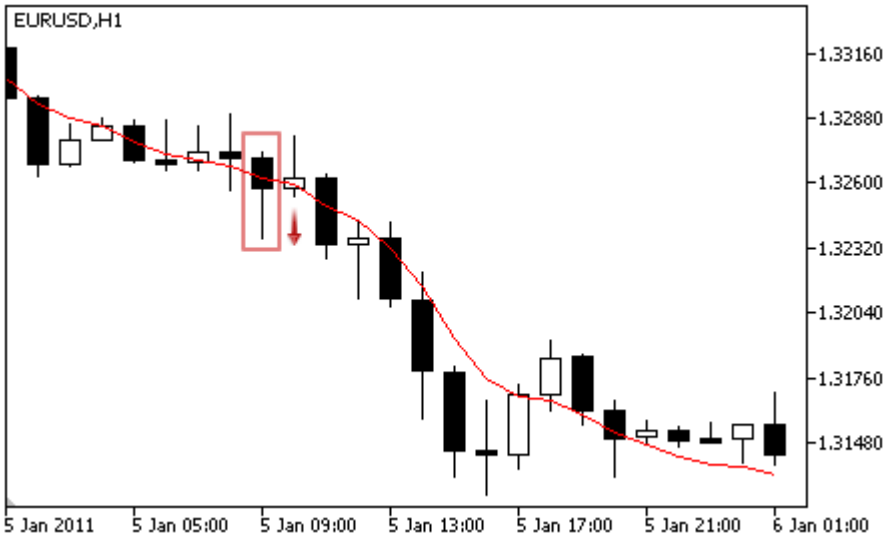
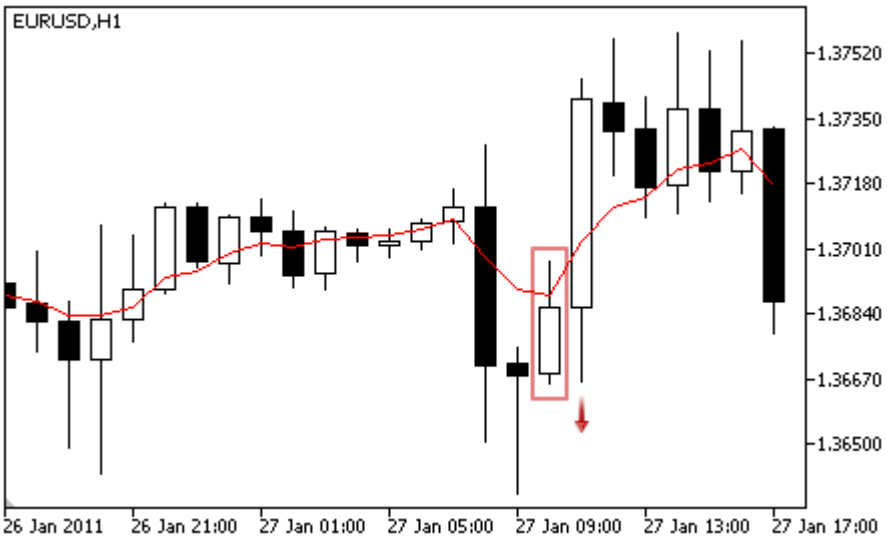
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para cima (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador aumenta (sinal forte). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• A sombra inferior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está acima do indicador, e a mínima do preço (Low) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco). 
Para Vender	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador diminui (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador diminui (sinal forte).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • A sombra superior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está abaixo do indicador, e a máxima do preço (High) está acima da linha do indicador) e o indicador diminui (sinal fraco). 
Sem objeções para comprar	O preço está acima do indicador.
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo da média do indicador.
Shift	Deslocamento do indicador ao longo do eixo do tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.

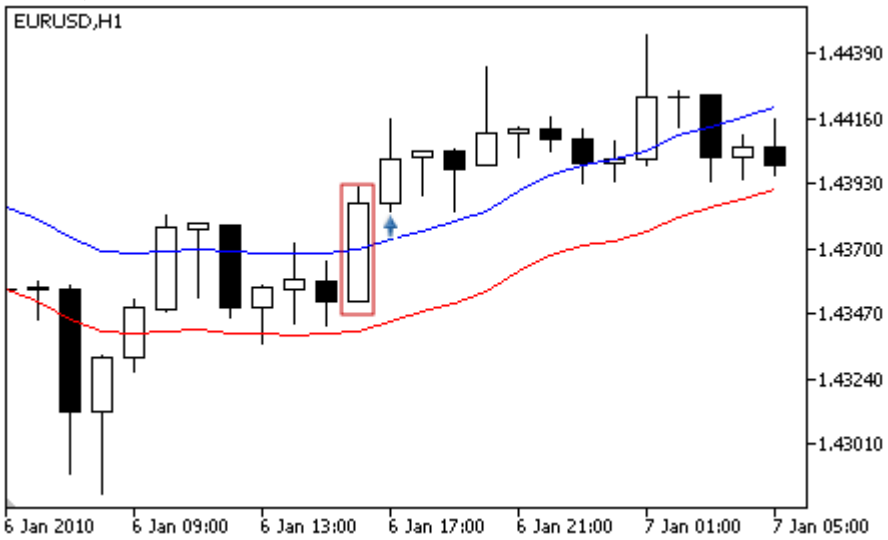
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Envelopes

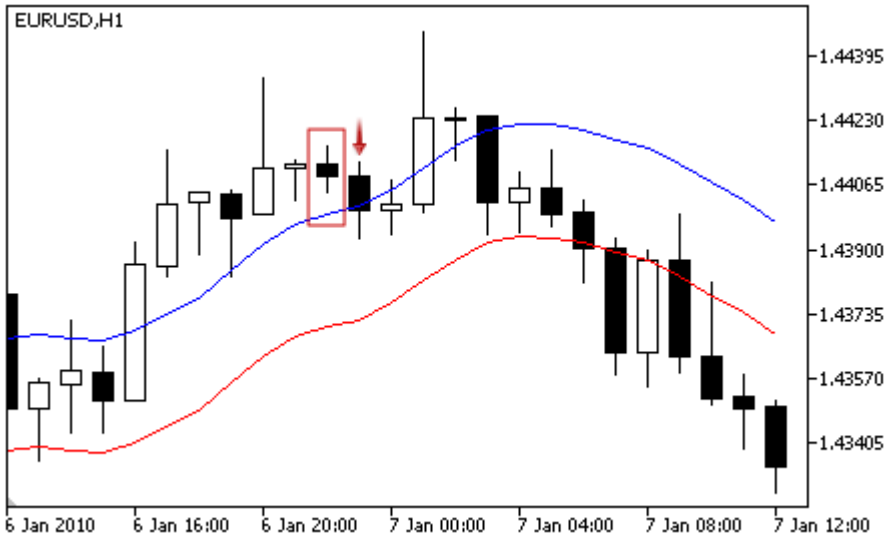
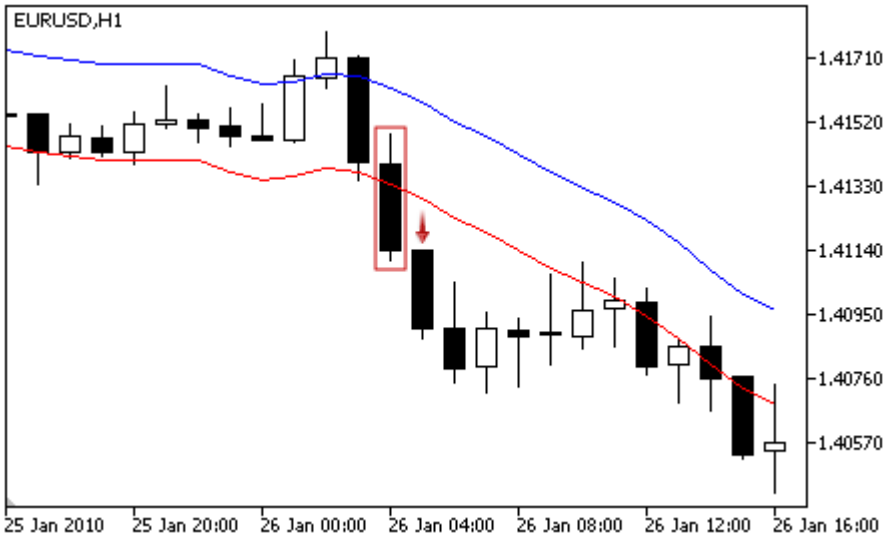
Este módulo de sinais baseia-se nos modelos de mercado do indicador Envelopes. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço está perto da linha inferior do indicador na barra analisada. • O preço cruzou a linha superior do indicador na barra analisada. 
Para Vender	• O preço é perto da linha superior do indicador na barra analisada.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • O preço cruzou a linha inferior do indicador na barra analisada. 
Sem objeções para comprar	Nenhum sinal.
Sem objeções para vender	Nenhum sinal.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo do indicador.
Shift	Deslocamento do indicador ao longo do eixo do tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.
Deviation	Desvio das bordas do envelope a partir da linha central (MA), em termos percentuais.

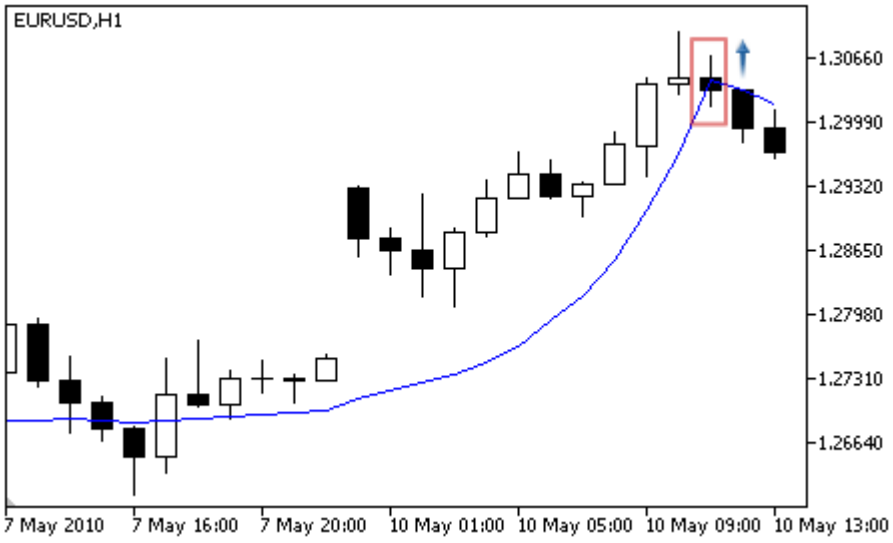
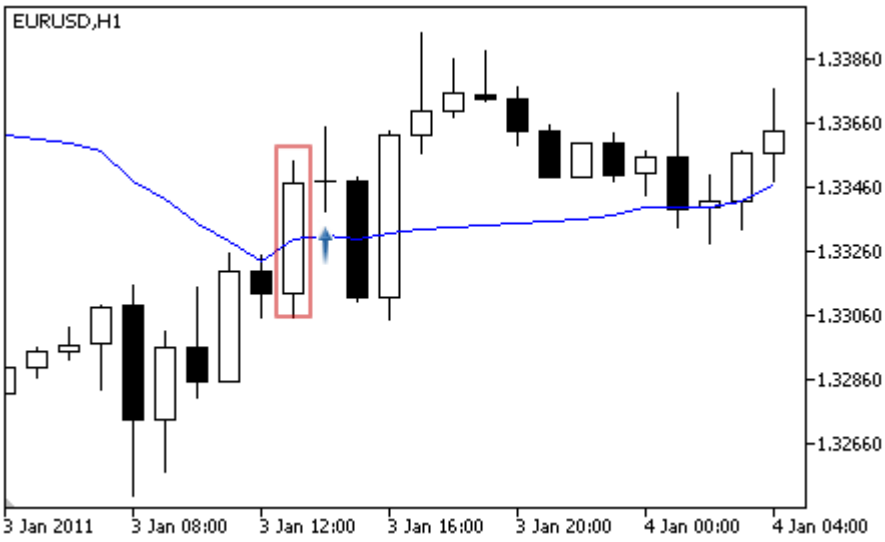
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Fractal Adaptive Moving Average

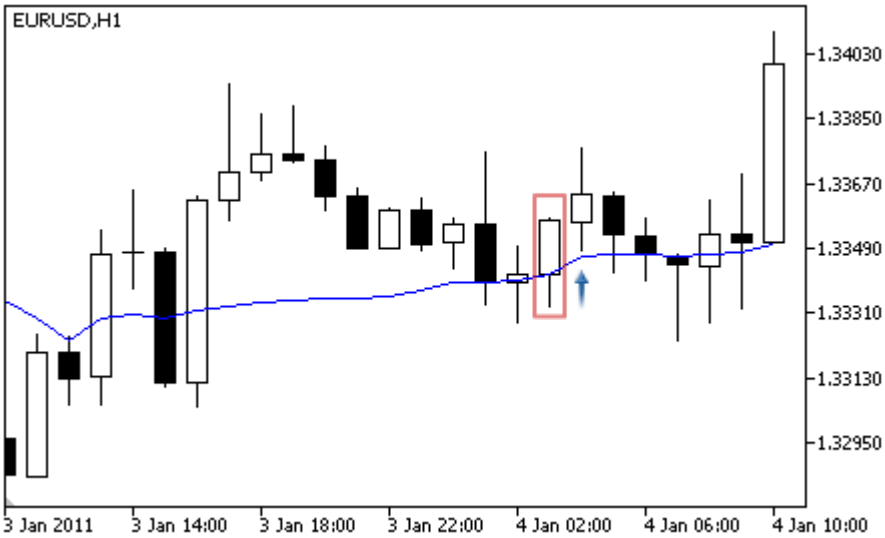
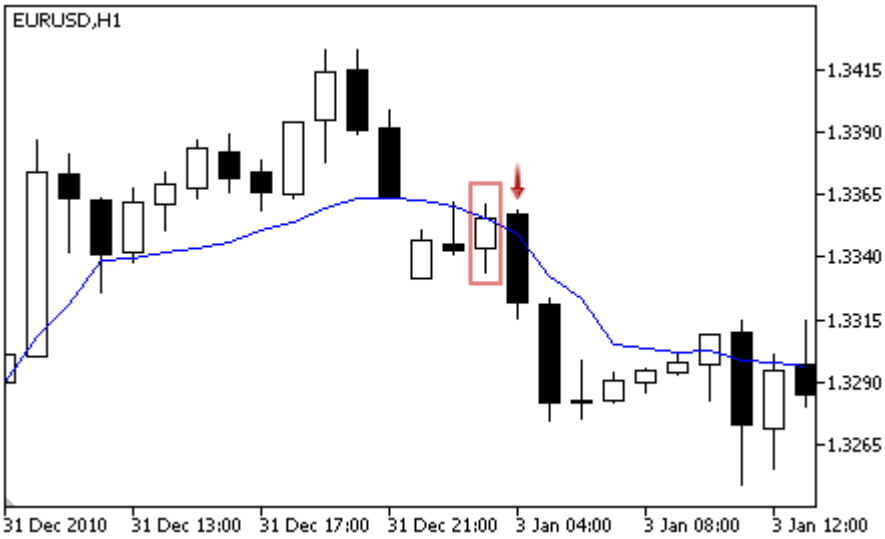
Este módulo de sinais baseia-se nos modelos de mercado do indicador Adaptive Fractal Média Móvel. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

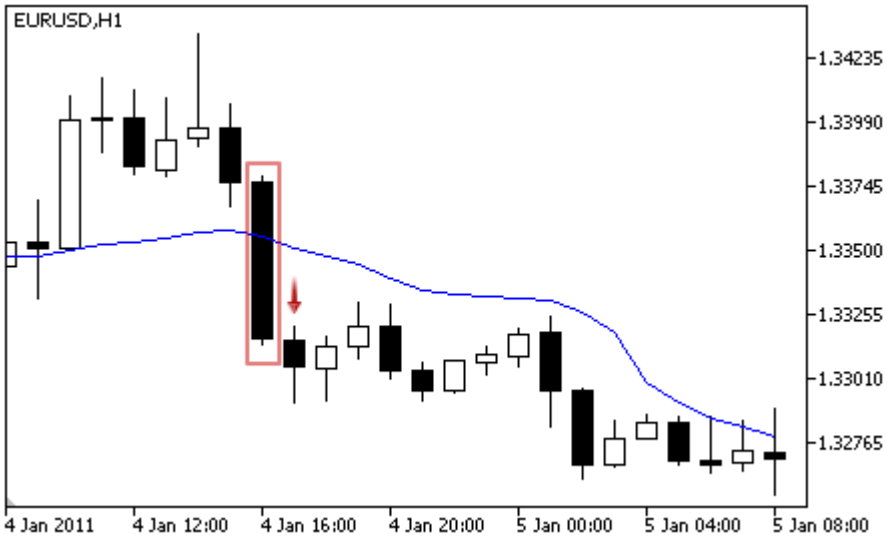
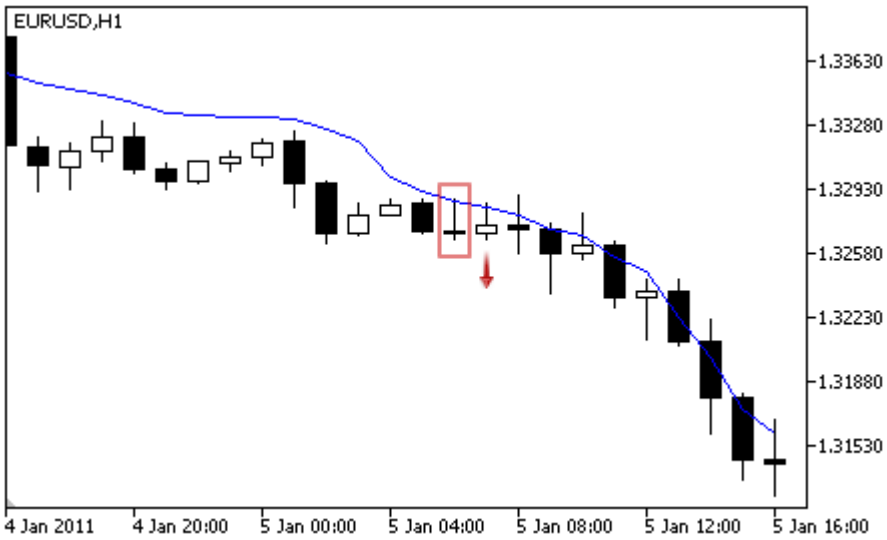
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para cima (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador aumenta (sinal forte). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• A sombra inferior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está acima do indicador, e a mínima do preço (Low) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco). 
Para Vender	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador diminui (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador diminui (sinal forte).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • A sombra superior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está abaixo do indicador, e a máxima do preço (High) está acima da linha do indicador) e o indicador diminui (sinal fraco). 
Sem objeções para comprar	Os preços estão acima do indicador.
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo da média do indicador.
Shift	Deslocamento do indicador ao longo do eixo do tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.

© 2000-2024, MetaQuotes Ltd.

Sinais do Filtro de Tempo Intradiário

Este módulo é baseado no pressuposto de que a eficiência dos modelos de mercado muda no decorrer do tempo. Usando este módulo, você pode filtrar sinais recebidos de outros módulos por horas e dias da semana. Isso permite aumentar a qualidade dos sinais gerados devido ao corte dos períodos de tempo desfavoráveis. O mecanismo das tomadas de decisões com base nos sinais dos módulos é descrito em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	Nenhum sinal.
Para Vender	Nenhum sinal.
Sem objeções para comprar	A data e a hora atual atendem os parâmetros especificados.
Sem objeções para vender	A data e a hora atual atendem os parâmetros especificados.

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
GoodHourOfDay	Hora do dia (de 0 a 23), quando os sinais de negociação serão habilitados. Se o valor é -1, os sinais serão habilitados por todo o dia.
BadHoursOfDay	O campo bit. Cada bit deste campo corresponde a uma hora do dia (bit 0 - 0 hora, ..., 23 bit - 23 horas). Se o valor de um bit é igual a 0, os sinais de negociação serão ativados durante a hora correspondente. Se o valor de um bit é igual a 1, os sinais de negociação serão desativados durante a hora correspondente. Um número específico é representado como um número binário e é usado como máscara de bit. Horas desabilitadas têm maior prioridade do que as habilitadas.
GoodDayOfWeek	Número de um único dia na semana (de 0 a 6, onde 0 é domingo), quando os sinais de negociação serão habilitados. Se o valor for -1, os sinais serão habilitados em qualquer dia.
BadDaysOfWeek	O campo bit. Cada bit deste campo corresponde a um dia da semana (bit 0 - Domingo, ..., 6 bit - sábado). Se o valor de um bit é igual a 0, os sinais de negociação serão ativados durante o dia correspondente. Se o valor de um bit é igual a 1, os sinais de negociação serão desativados durante o dia correspondente. Um número específico é representado como um número binário e é usado como máscara de bit.

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
	Dias desabilitados têm maior prioridade do que os habilitados.

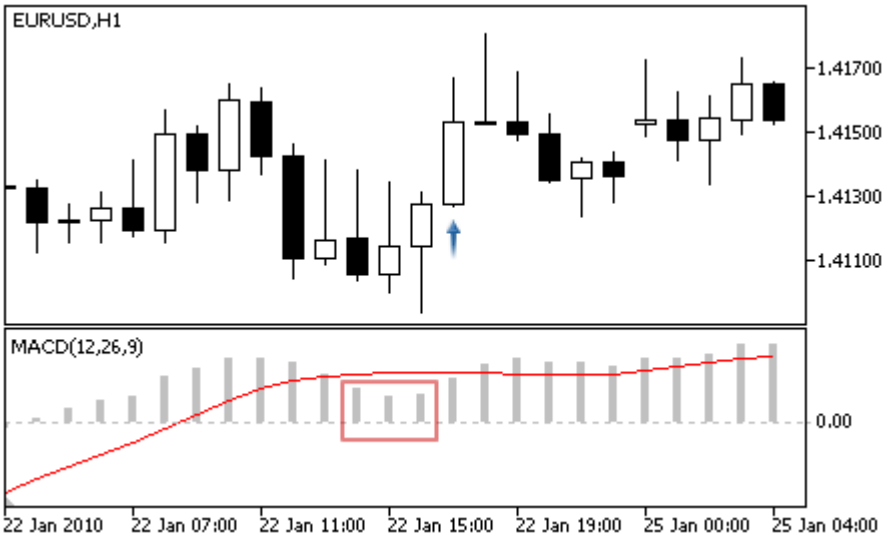
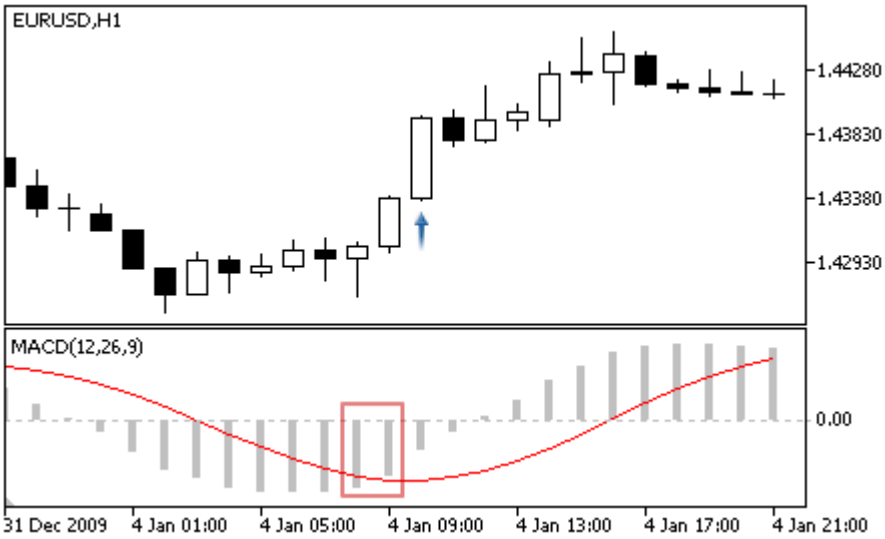
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador MACD

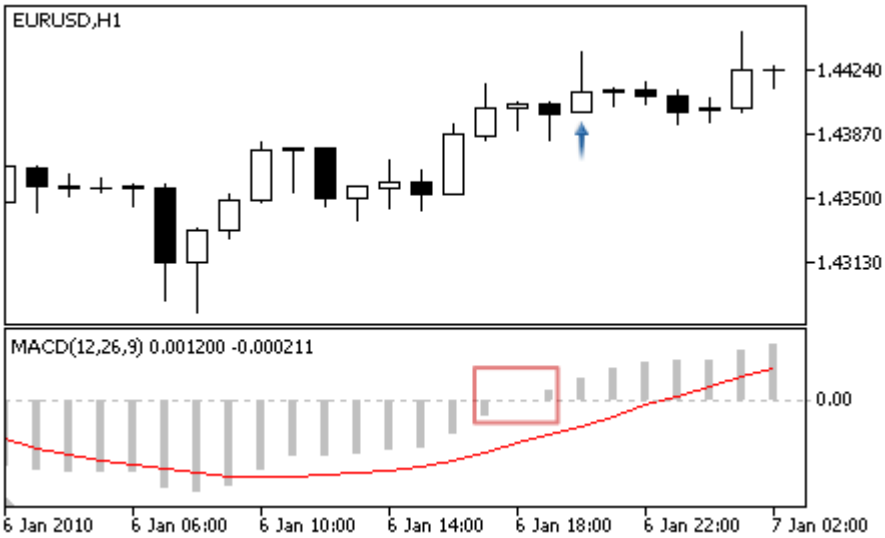
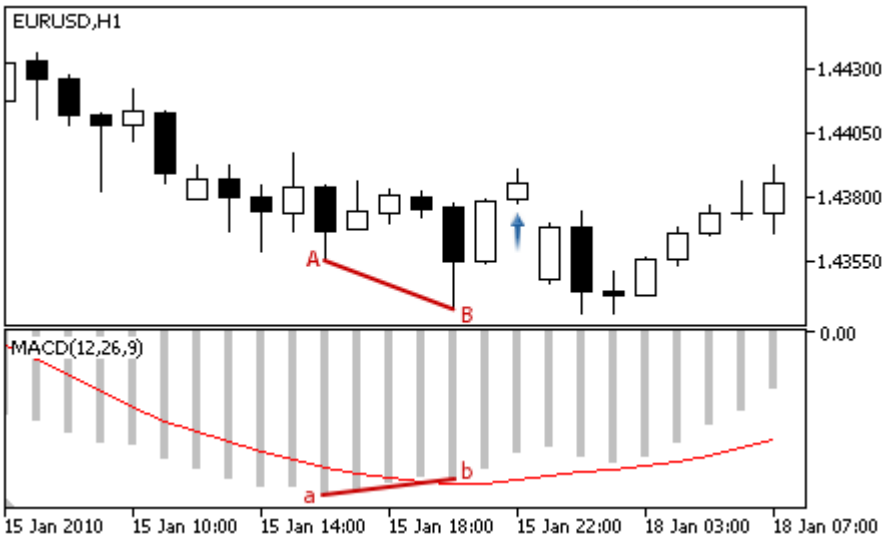
Este módulo de sinais baseia-se nos modelos de mercado do oscilador MACD. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

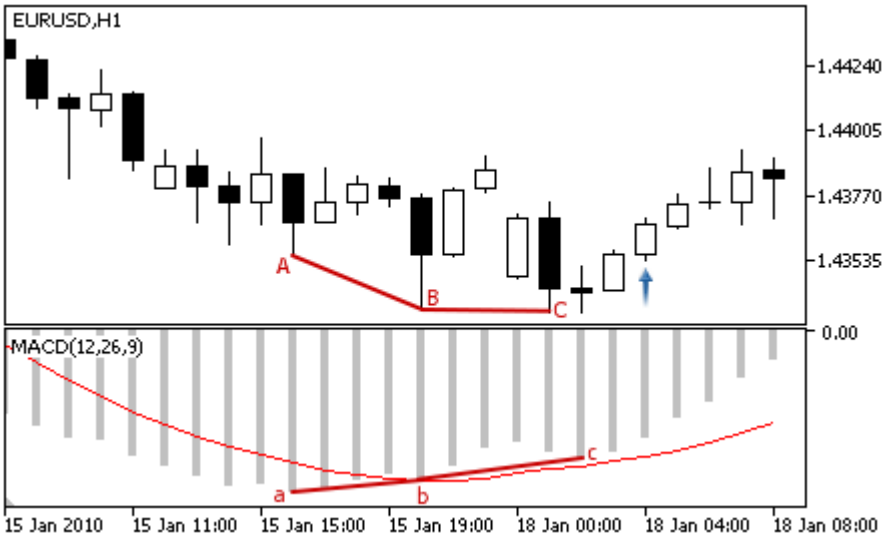
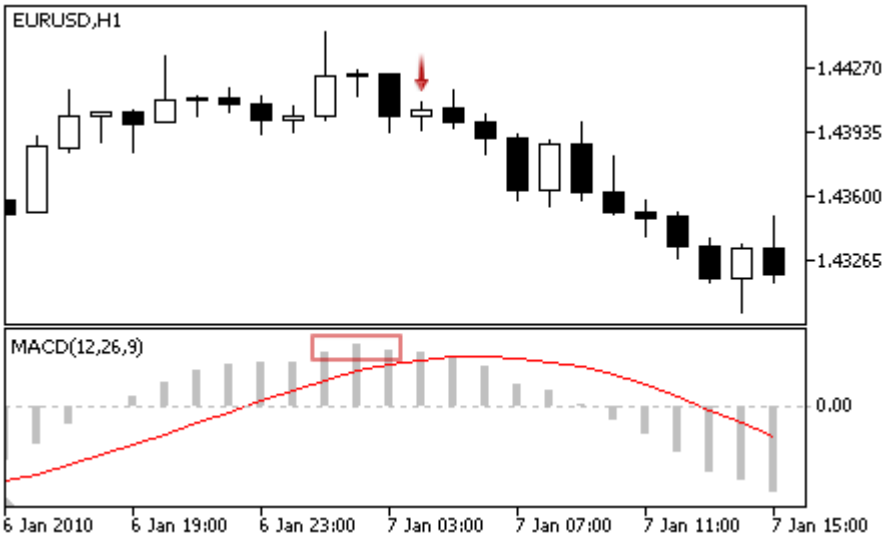
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão** - O oscilador se volta para cima (o oscilador cresce na barra analisada e decresce na anterior).  • **Cruzamento da linha principal e de sinal** - A linha principal está acima da linha de sinal para a barra analisada e abaixo da linha do sinal na barra anterior.  • **Cruzando o nível de zero** - A linha principal está acima do nivel zero no barra analisada e abaixo do nível zero na anterior.

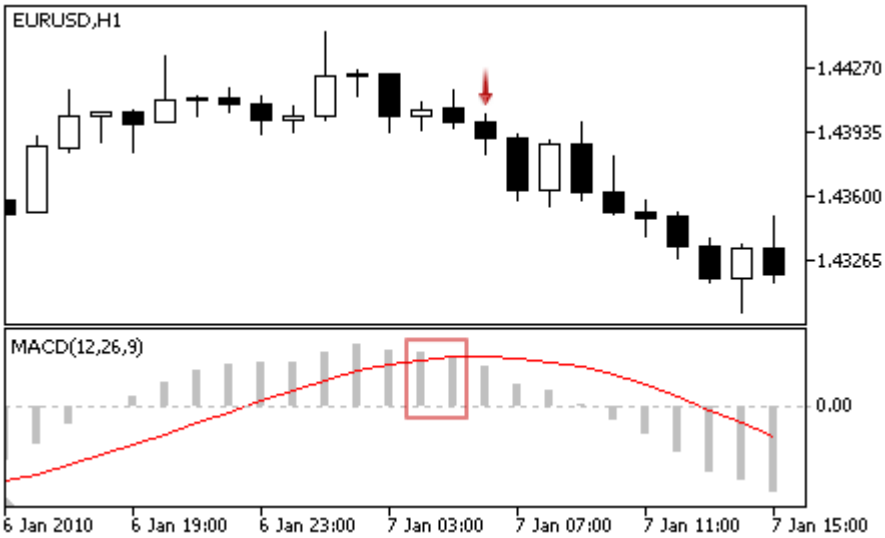
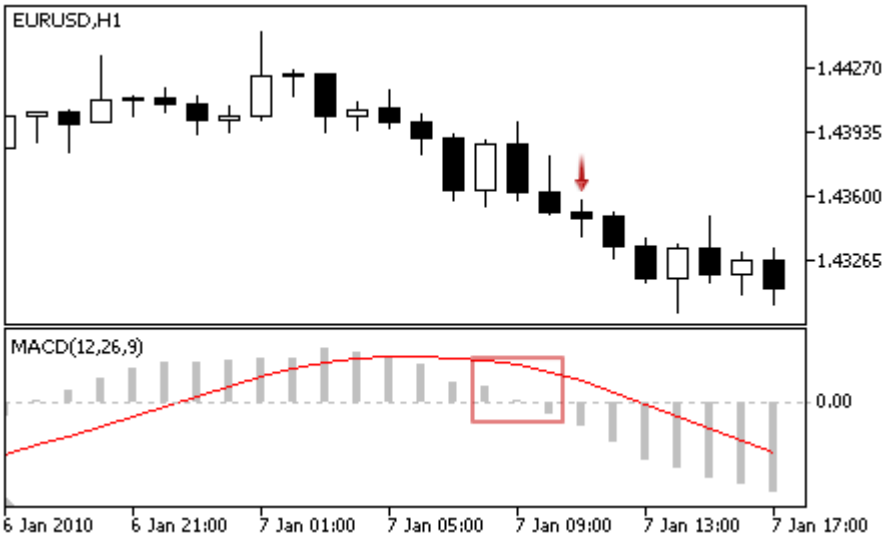
© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B).  • **Divergência Dupla** - a linha do oscilador forma três consequentes fundos (a/b/c), onde o valor de a < b < c; correspondendo a três mínimas (Low) de preço das barras formadas (A/B/C), onde o valor da mínima do preço A > B > C.

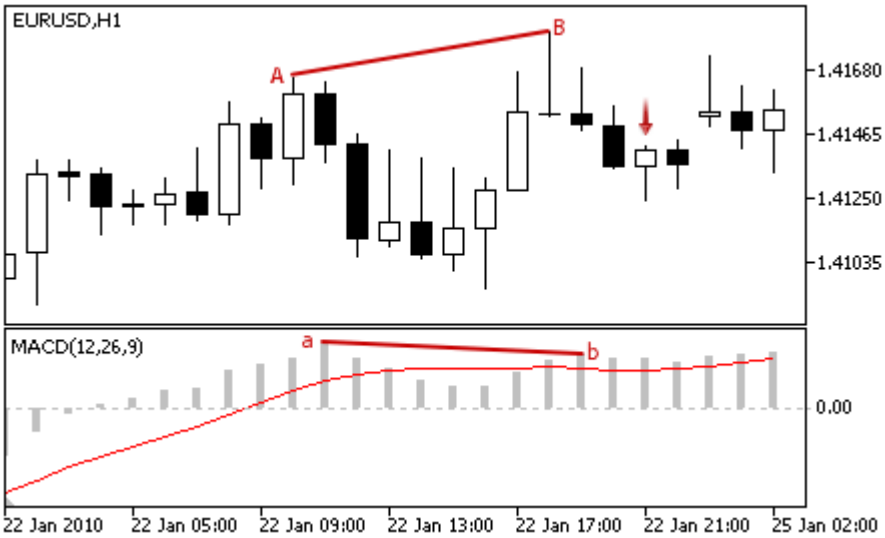
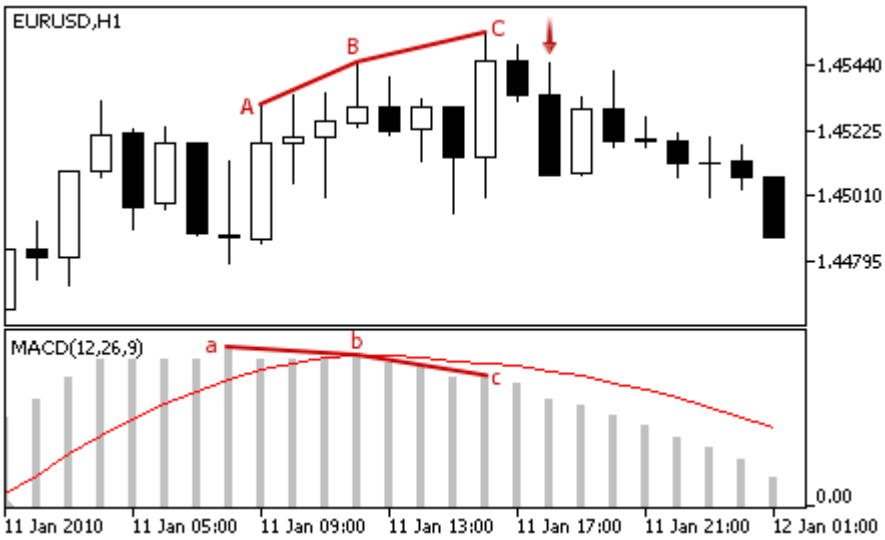
© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	
Para Vender	• **Reversão** - O oscilador se volta para baixo (o oscilador decresce na barra analisada e cresce na barra anterior).  • **Cruzamenot da linha principal e de sinal** - A linha principal está abaixo da linha de sinal na barra analisada e acima da linha de sinal na barra anterior.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Cruzando o nível de zero** - A linha principal está abaixo do nível zero na barra analisada e acima do nível zero no anterior.  • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência Dupla** - a linha do oscilador forma três consequentes topos (a/b/c), onde o valor de a > b > c; correspondendo a três máximas (High) de preço das barras formadas (A/B/C), onde o valor da máxima do preço A < B < C. 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodFast	Período de cálculo da EMA rápida.
PeriodSlow	Período de cálculo da EMA lenta.
PeriodSignal	Período de suavização.
Applied	As séries de preços utilizadas para o cálculo do oscilador.

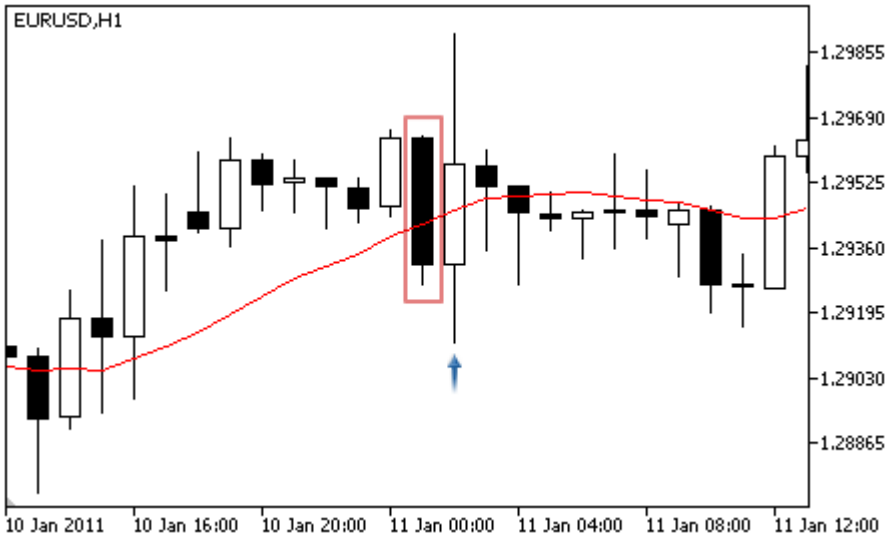
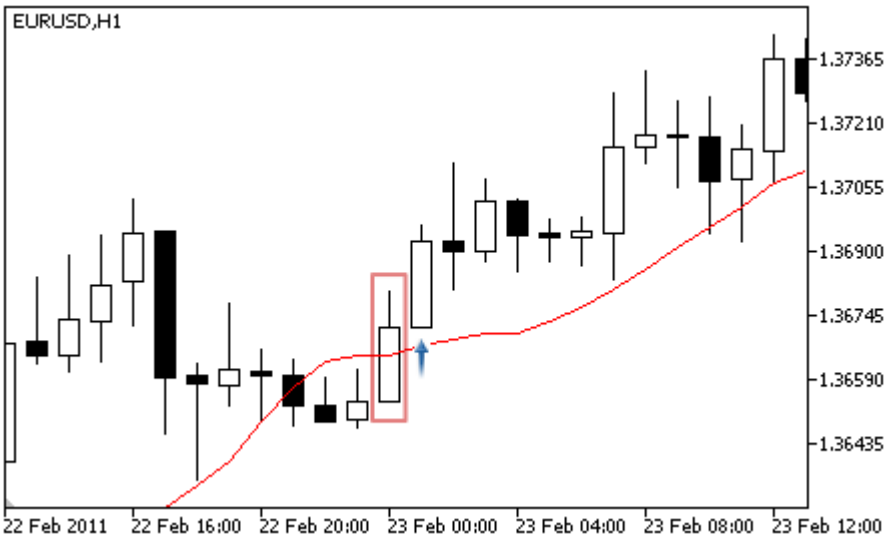
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Moving Average

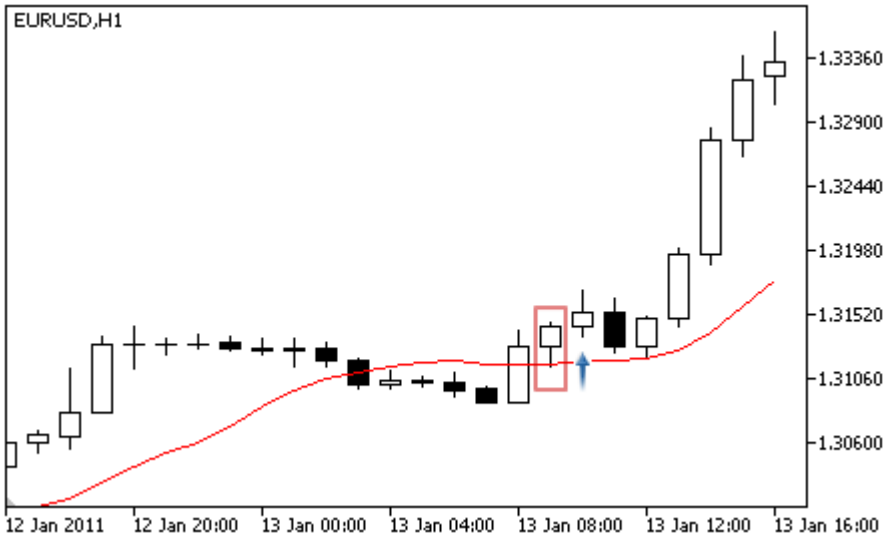
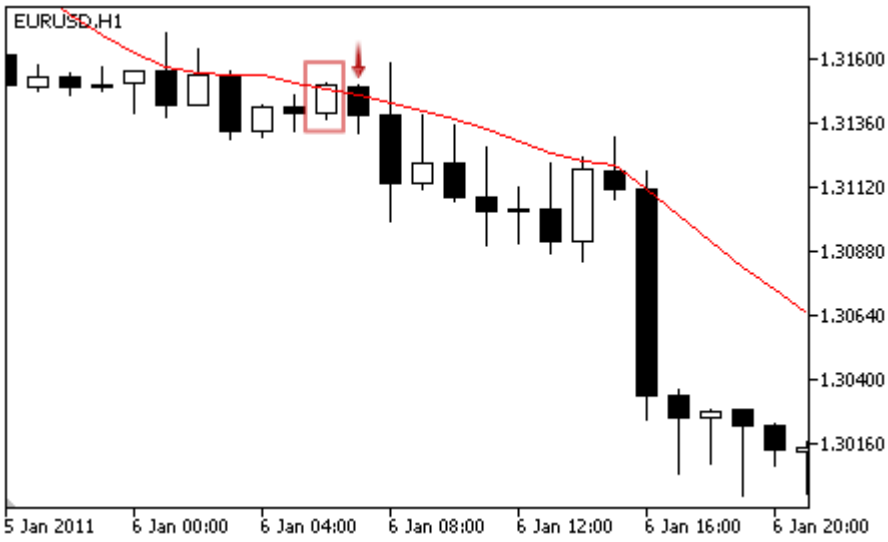
Este módulo de sinais baseia-se nos modelos de mercado do indicador Média Móvel. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

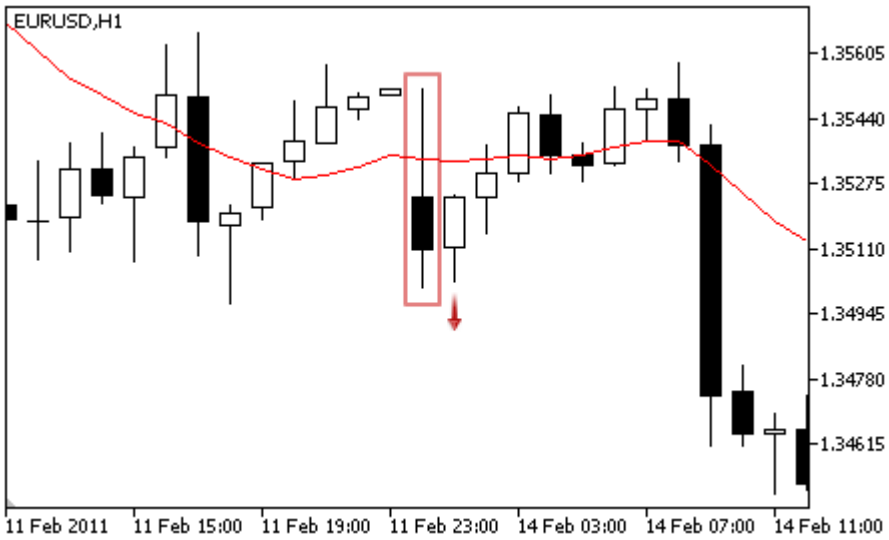
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para cima (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador aumenta (sinal forte). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• A sombra inferior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está acima do indicador, e a mínima do preço (Low) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco). 
Para Vender	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador diminui (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador diminui (sinal forte).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • A sombra superior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está abaixo do indicador, e a máxima do preço (High) está acima da linha do indicador) e o indicador diminui (sinal fraco). 
Sem objeções para comprar	Os preços estão acima do indicador.
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo da média do indicador.
Shift	Deslocamento do indicador ao longo do eixo do tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.

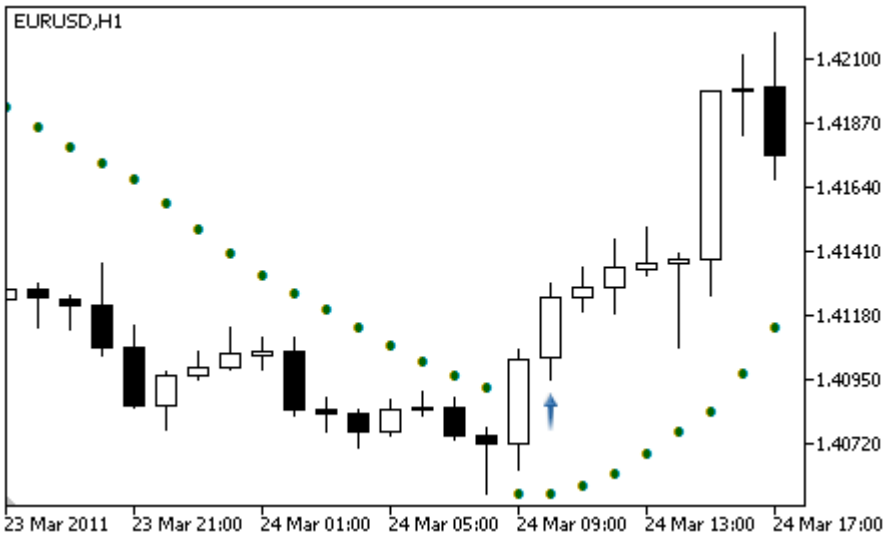
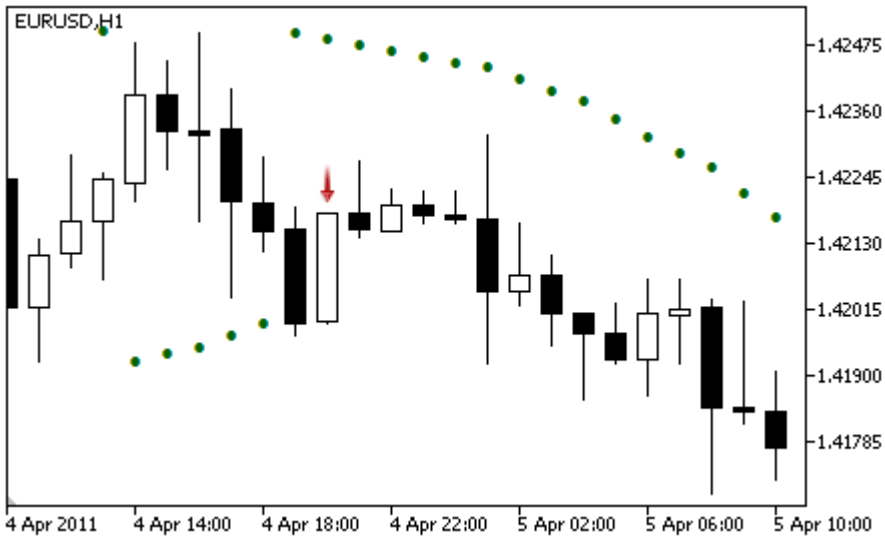
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Parabolic SAR

Este módulo de sinais baseia-se nos modelos de mercado do indicador Parabólico SAR. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	**Reversão** — O indicador está abaixo do preço na barra analisada e acima do preço na anterior. 
Para Vender	**Reversão** — O indicador está acima do preço na barra analisada e abaixo do preço na anterior. 
Sem objeções para comprar	Os preços estão acima do indicador.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
Step	O aumento da velocidade do indicador.
Maximum	Taxa máxima da velocidade de convergência do indicador com o preço.

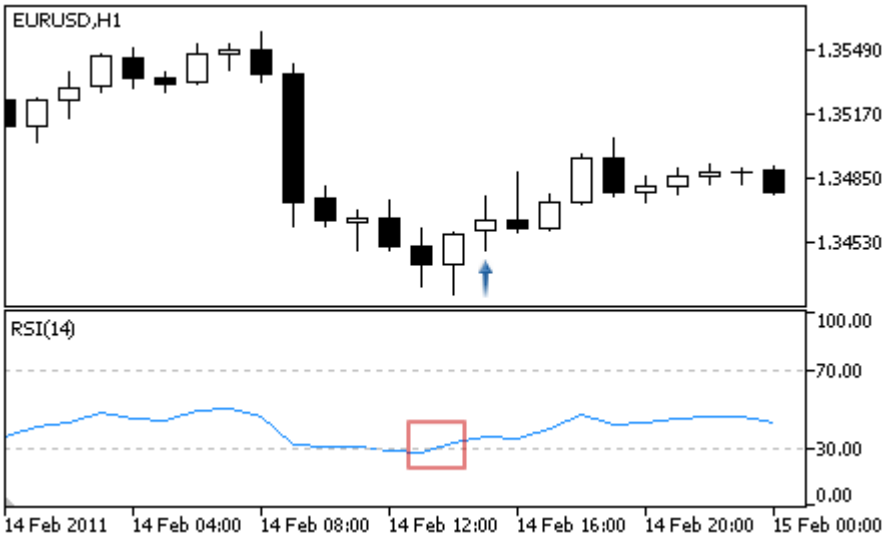
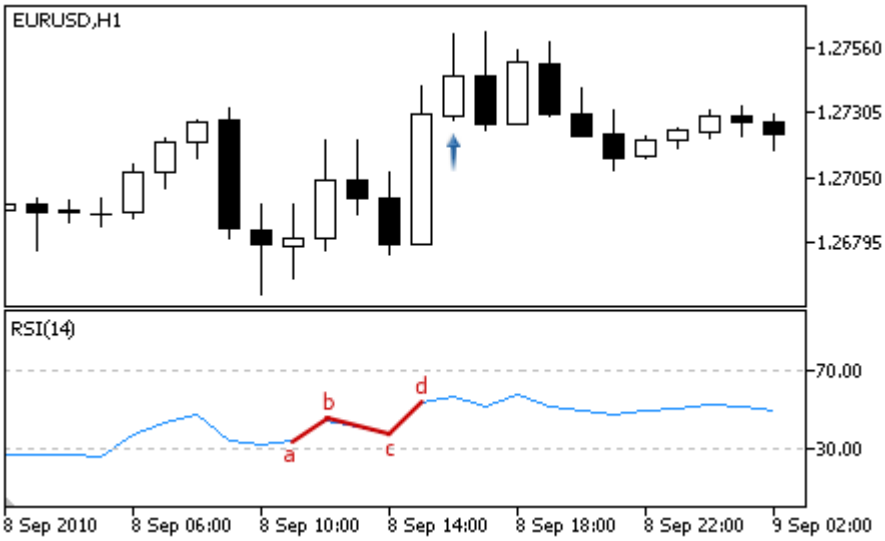
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador Relative Strength Index

Este módulo de sinais baseia-se nos modelos de mercado do oscilador Relative Strength Index. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

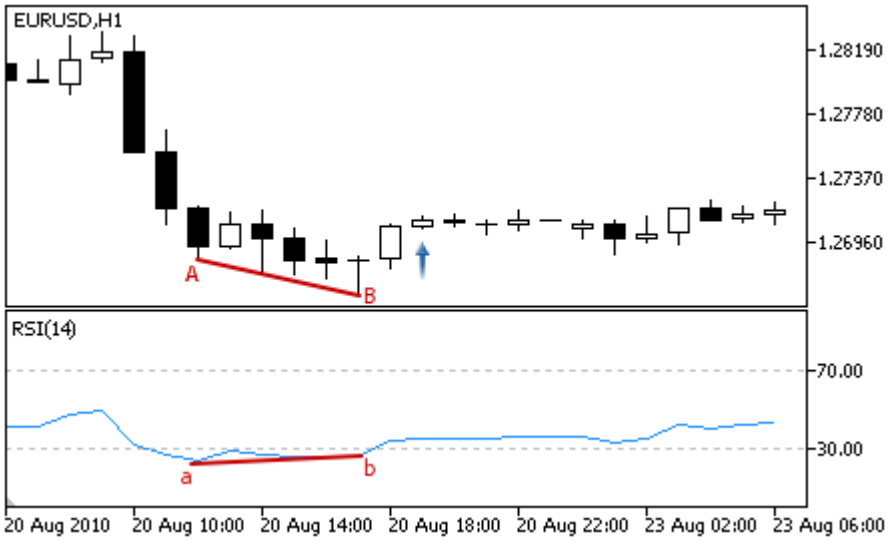
Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

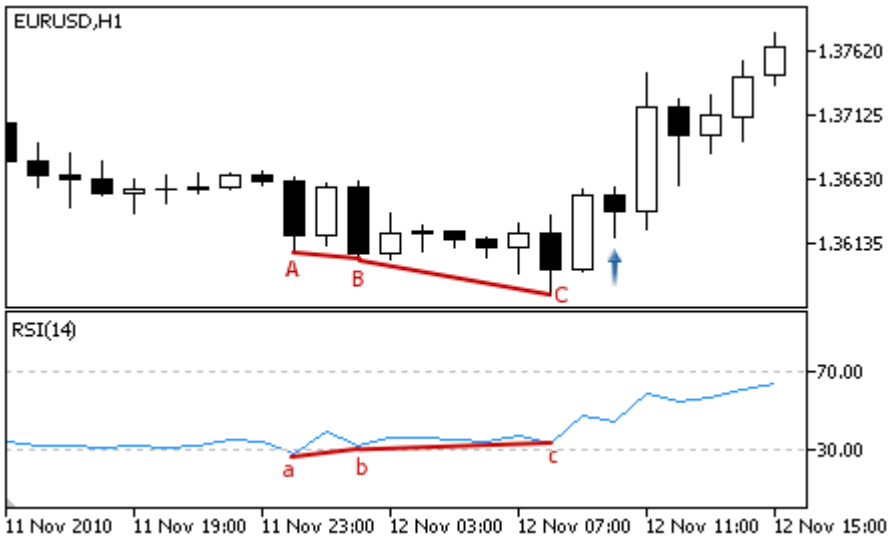
Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão atrás do nível sobrevendido** - O oscilador se volta para cima e seu valor na barra analisada está atrás do nível sobrevendido (valor padrão é 30).  • **Falha do Swing** - o oscilador cresce mais alto do que o pico anterior na barra analisada.  • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B).

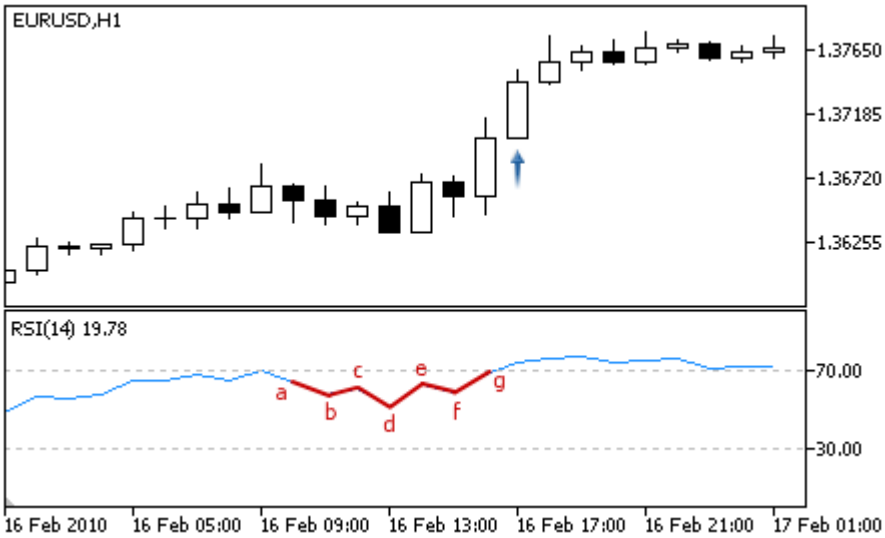
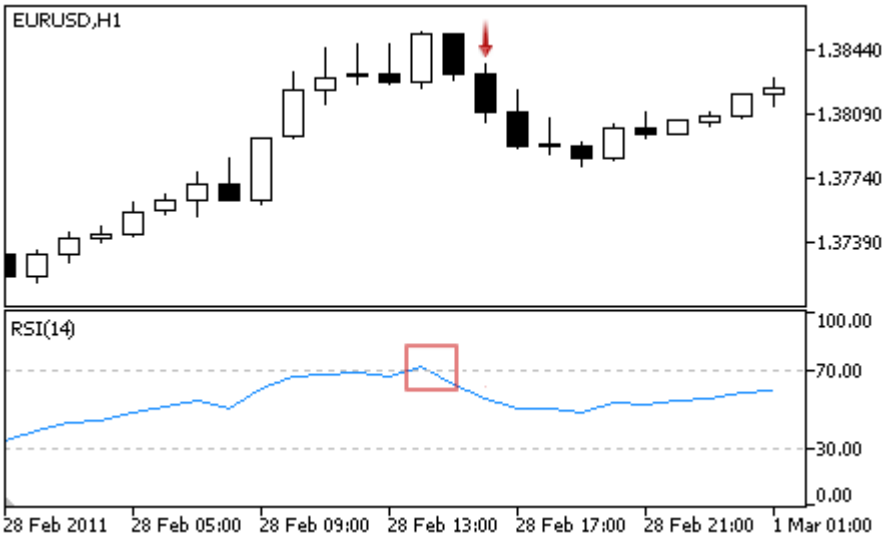


- **Divergência Dupla** - a linha do oscilador forma três consequentes fundos (a/b/c), onde o valor de a < b < c; correspondendo a três mínimas (Low) de preço das barras formadas (A/B/C), onde o valor da mínima do preço A > B > C.

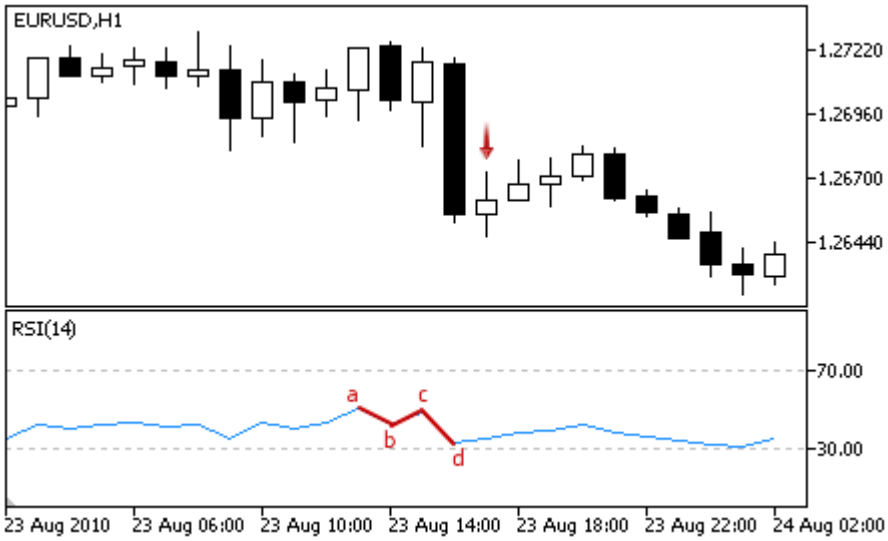
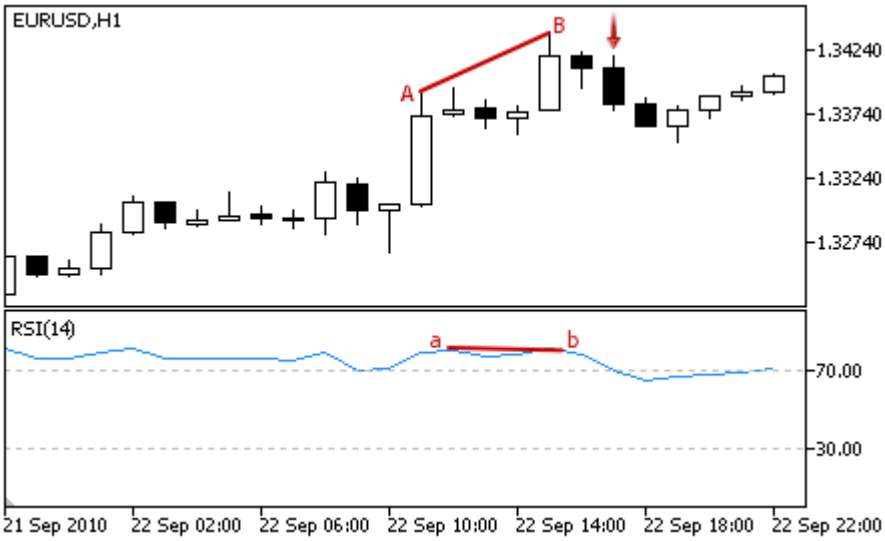


- **Cabeça/Ombros** — O oscilador forma três consequentes fundos, onde o central é um valor menor do que os outros.

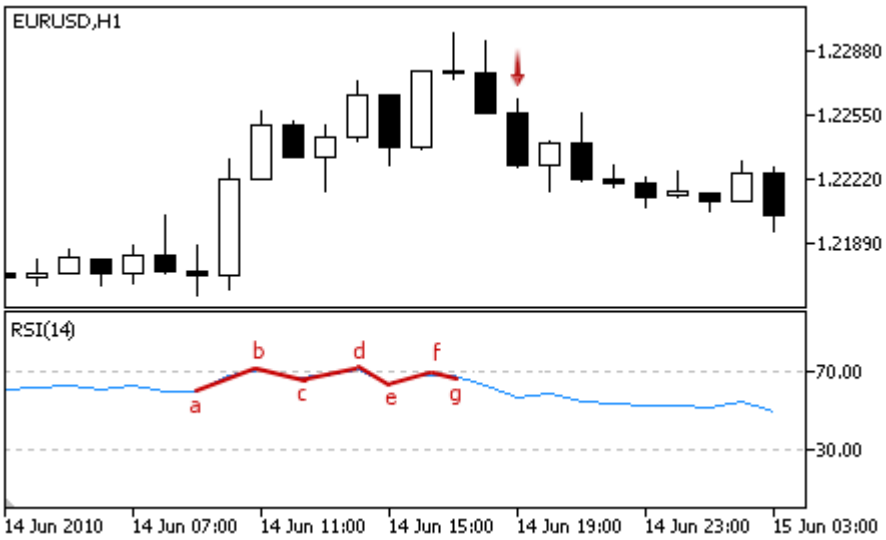
© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	
Para Vender	• **>Reversão atrás do nível sobrevendido** - O oscilador se volta para baixo e seu valor na barra analisada está por trás do nível de sobrecomprado (valor padrão é 70).  • **Falha do swing** - o oscilador decresce mais baixo do que o fundo anterior na barra analisada.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B).  • **Divergência Dupla** - a linha do oscilador forma três consequentes topos (a/b/c), onde o valor de a > b > c; correspondendo a três máximas (High) de preço das barras formadas (A/B/C), onde o valor da máxima do preço A < B < C.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Cabeça/Ombros** — O oscilador forma três consequentes topos, onde o central é um valor maior do que os outros. 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodRSI	Período de cálculo do oscilador.
Applied	As séries de preços utilizadas para o cálculo do oscilador.

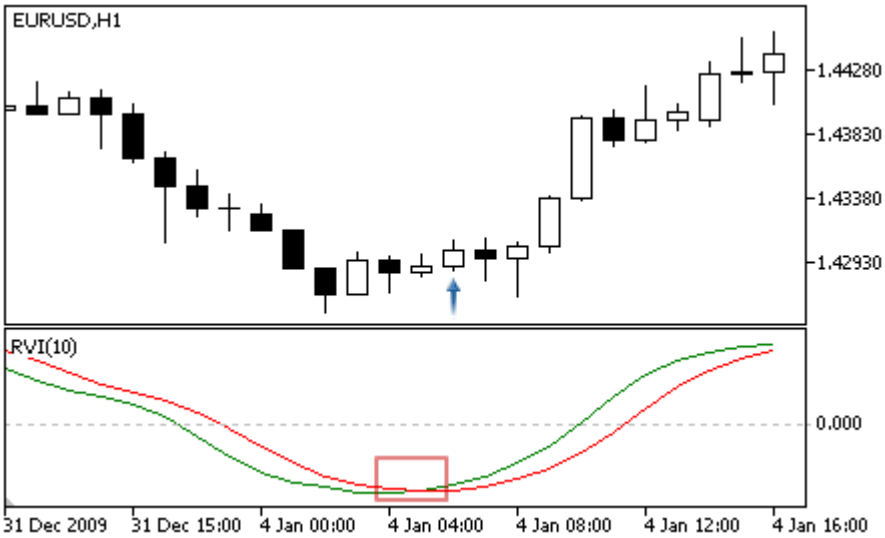
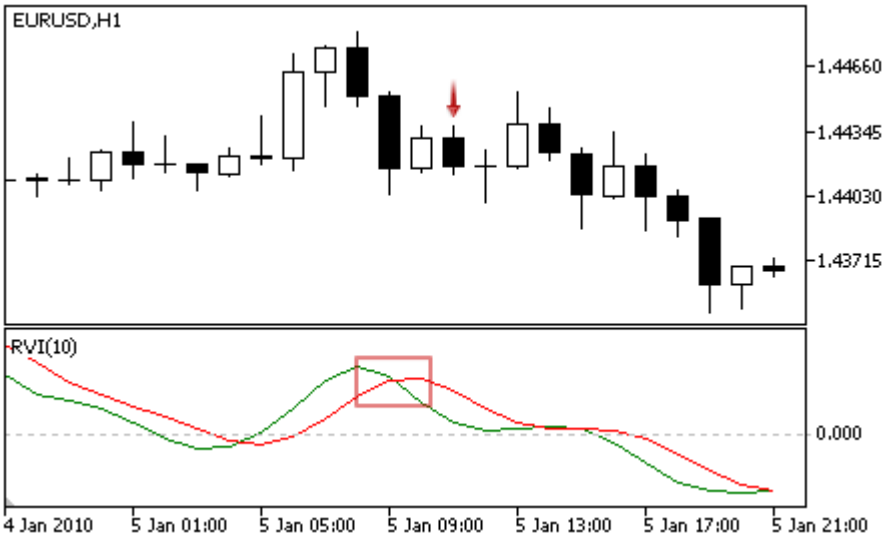
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador Relative Vigor Index

Este módulo de sinais baseia-se nos modelos de mercado do oscilador Relativa Índice Vigor. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	**Cruzamento da linha principal e de sinal** - A linha principal está acima da linha de sinal em relação a barra analisada e abaixo da linha do sinal na anterior. 
Para Vender	**Cruzamento da linha principal e de sinal** - A linha principal está abaixo da linha de sinal em relação a barra analisada e acima da linha do sinal na anterior. 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodRVI	Período de cálculo do oscilador.

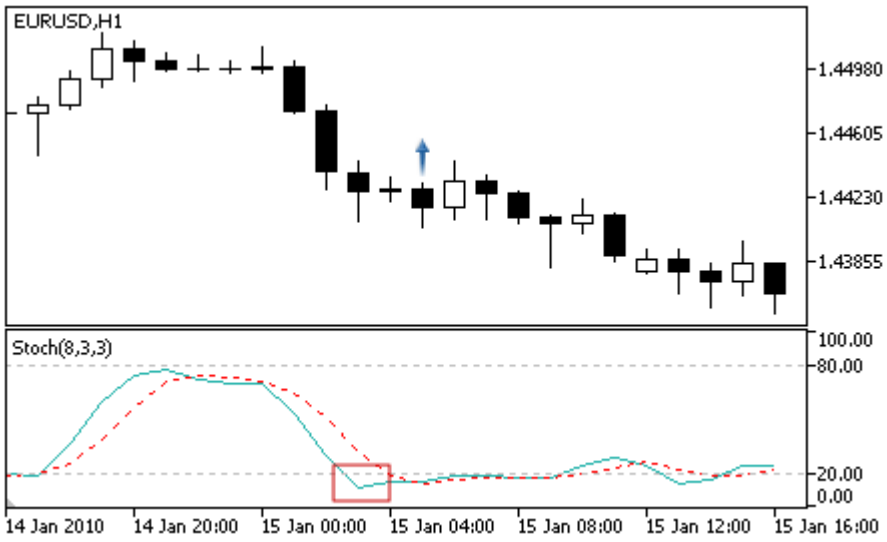
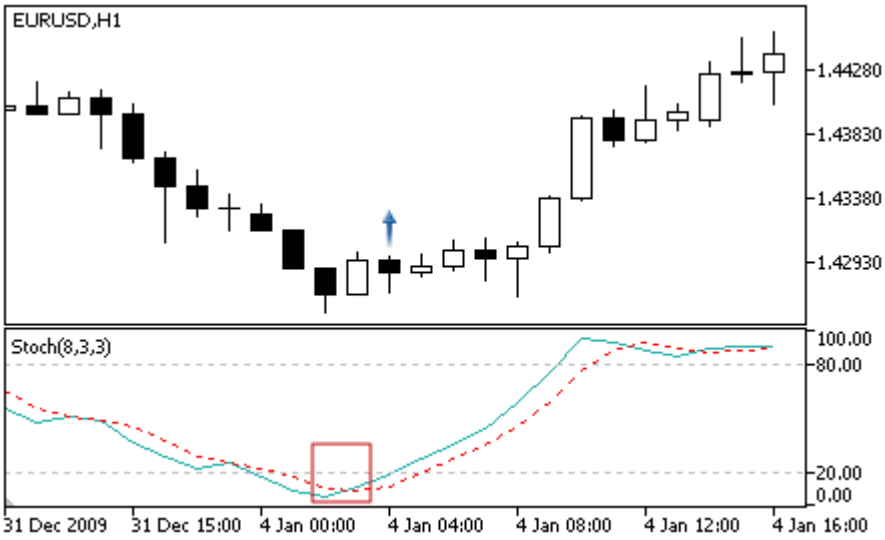
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador Stochastic

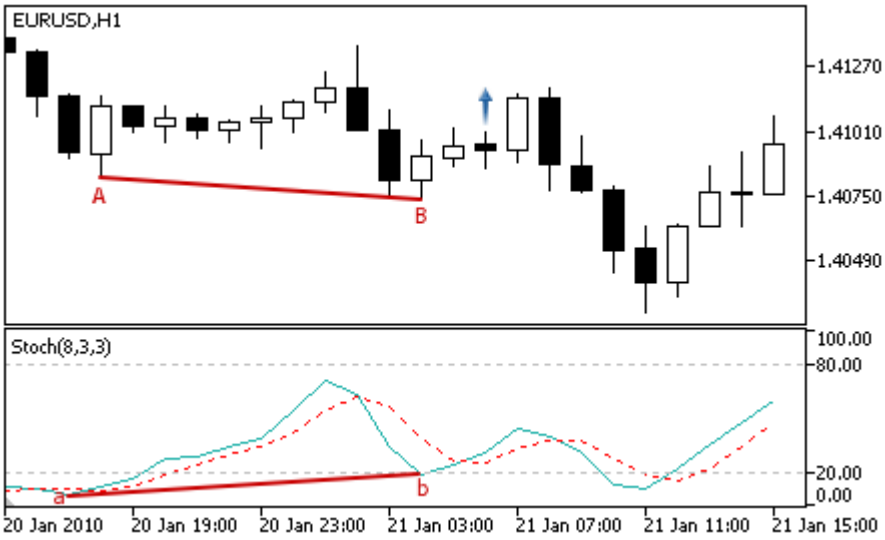
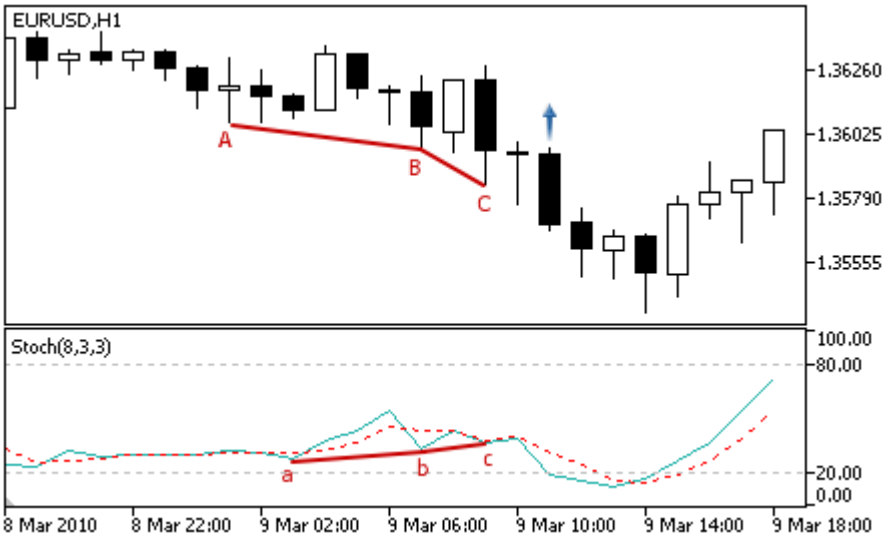
Este módulo de sinais com base nos modelos de mercado do oscilador Stochastic. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

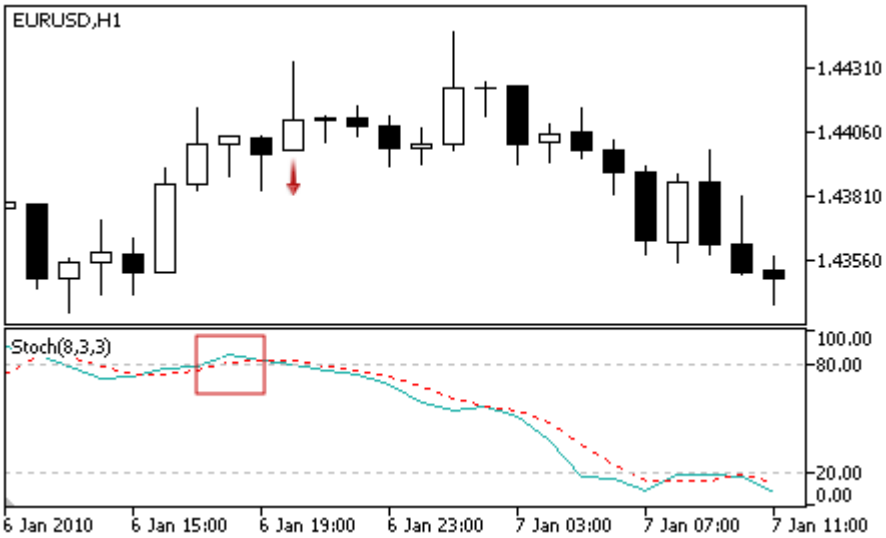
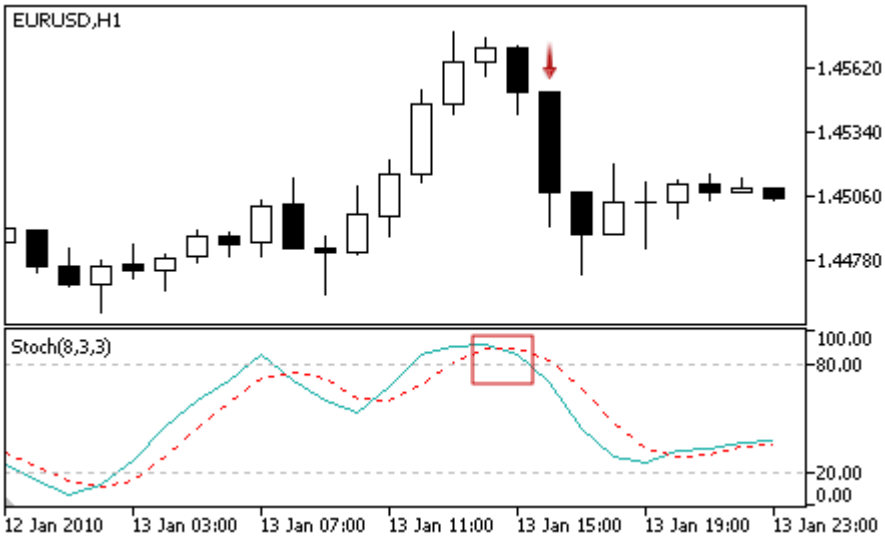
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão** - O oscilador se volta para cima (o oscilador cresce na barra analisada e decresce na anterior).  • **Cruzamento da linha principal e de sinal** - A linha principal está acima da linha de sinal em relação a barra analisada e abaixo da linha do sinal na anterior. 

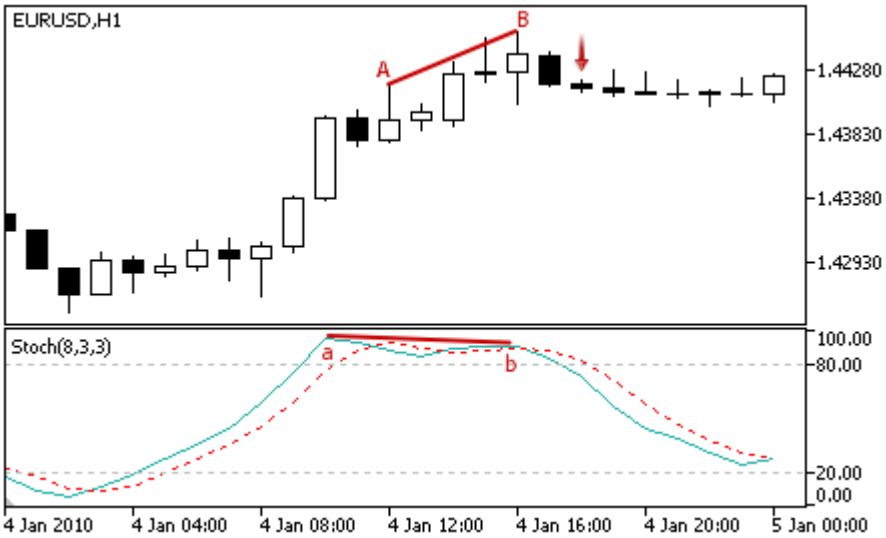
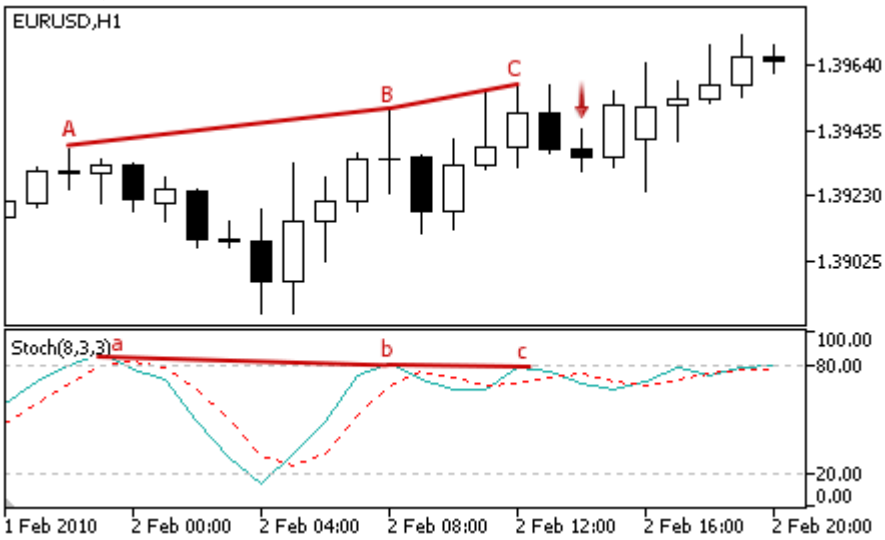
© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B).  • **Divergência Dupla** - a linha do oscilador forma três consequentes fundos (a/b/c), onde o valor de a < b < c; correspondendo a três mínimas (Low) de preço das barras formadas (A/B/C), onde o valor da mínima do preço A > B > C. 
Para Vender	• **Reversão** - O oscilador se volta para baixo (o oscilador decresce na barra analisada e cresce na barra anterior).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Cruzamento da linha principal e de sinal** - A linha principal está abaixo da linha de sinal em relação a barra analisada e acima da linha do sinal na anterior.  • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência Dupla** - a linha do oscilador forma três consequentes topos (a/b/c), onde o valor de a > b > c; correspondendo a três máximas (High) de preço das barras formadas (A/B/C), onde o valor da máxima do preço A < B < C. 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodK	Período de cálculo da linha principal do oscilador.
PeriodD	Período de cálculo da linha principal do oscilador.
PeriodSlow	Período de desaceleração.
Applied	As séries de preços utilizadas para o cálculo do oscilador.

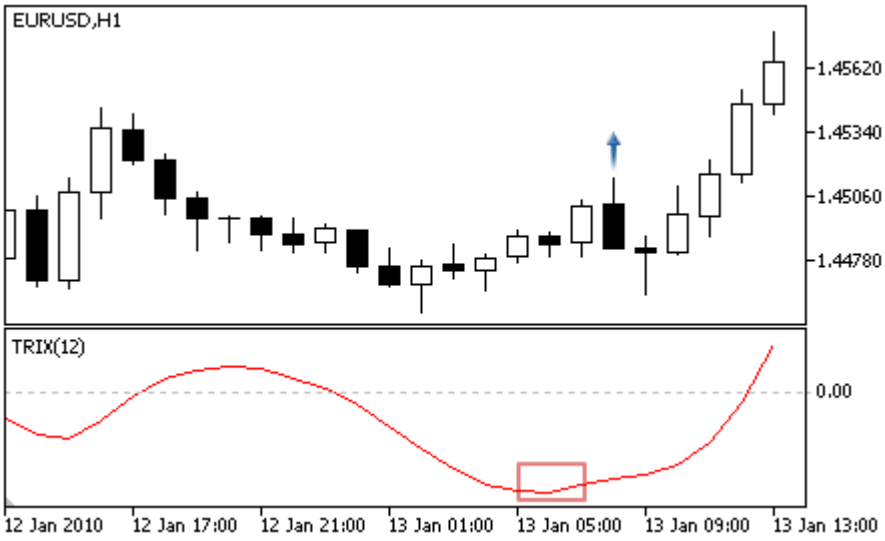
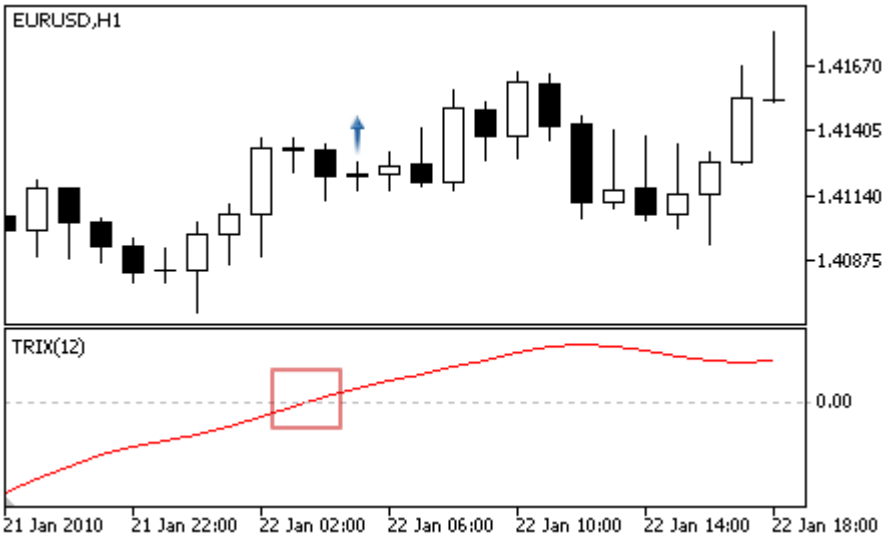
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador Triple Exponential Average

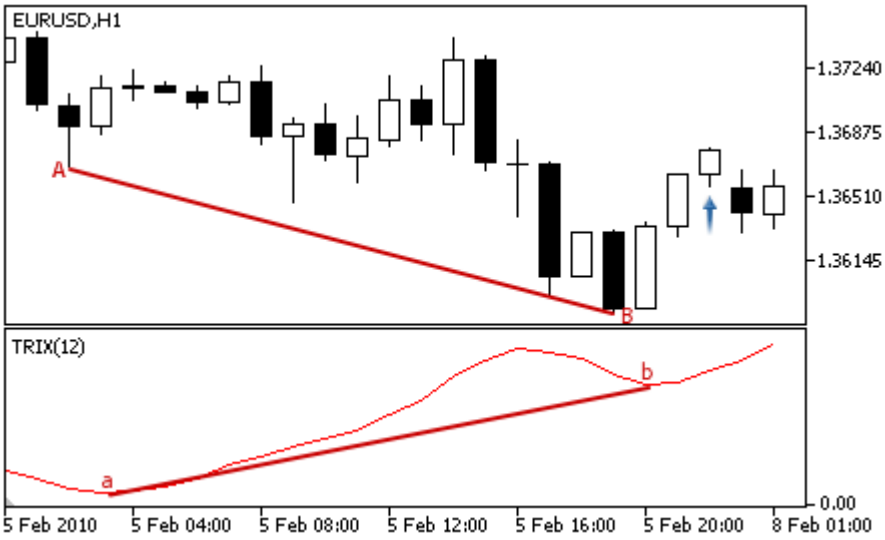
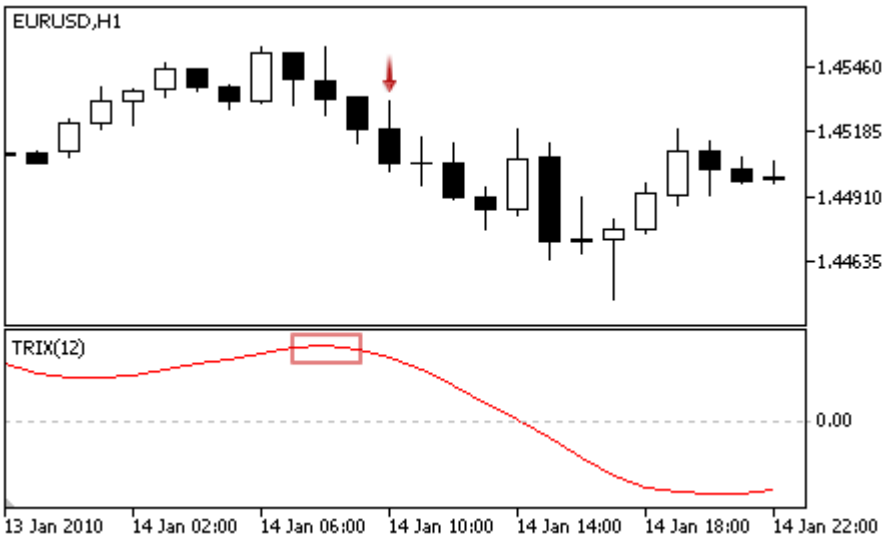
Este módulo de sinais baseia-se nos modelos do mercado do oscilador Triple Exponential Average. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

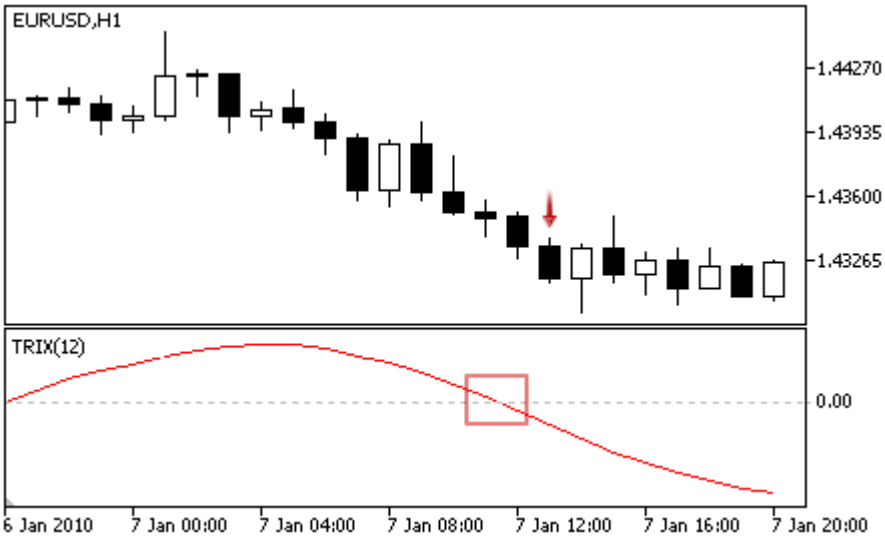
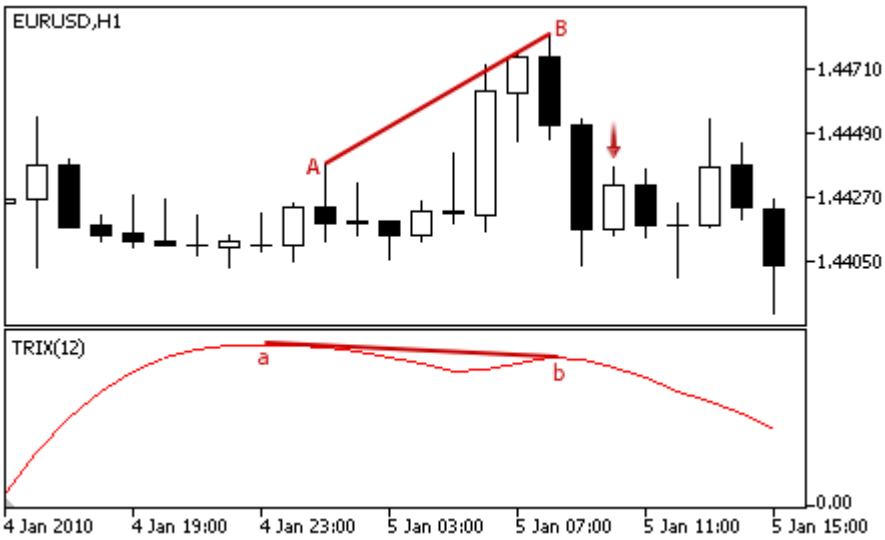
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão** - O oscilador se volta para cima (o oscilador cresce na barra analisada e decresce na anterior).  • **Cruzando o nível de zero** - A linha principal está acima do nivel zero no barra analisada e abaixo do nível zero na anterior.  • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B). 
Para Vender	• **Reversão** - O oscilador se volta para baixo (o oscilador decresce na barra analisada e cresce na barra anterior).  • **Cruzando o nível de zero** - A linha principal está abaixo do nível zero na barra analisada e acima do nível zero no anterior.

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B). 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodTriX	Período de cálculo do oscilador.
Applied	As séries de preços utilizadas para o cálculo do oscilador.

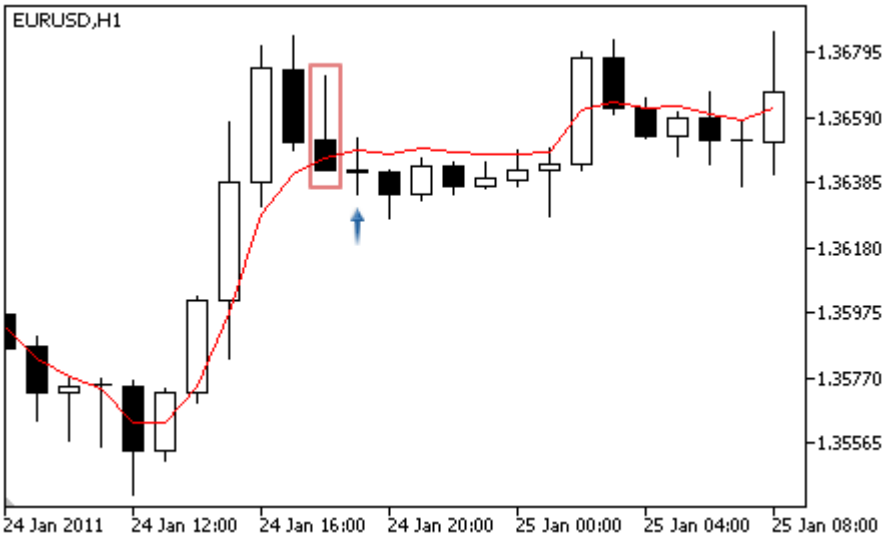
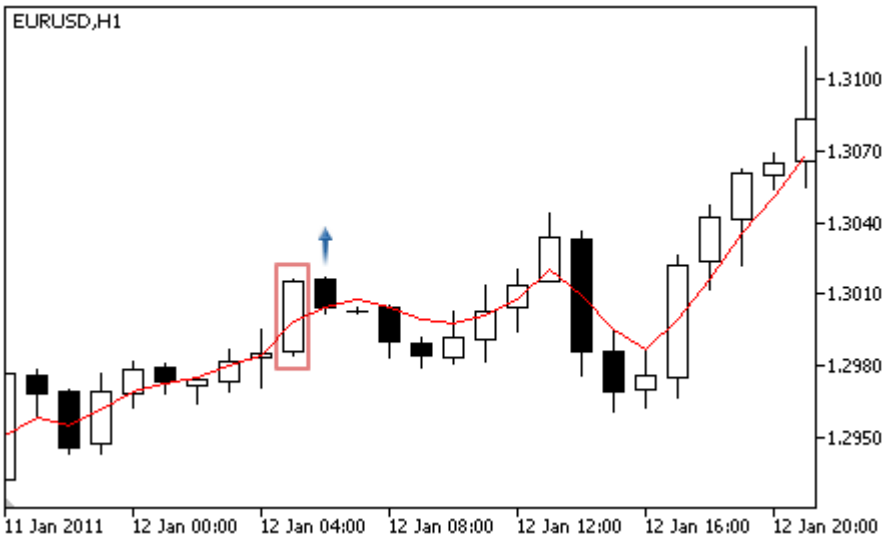
© 2000-2024, MetaQuotes Ltd.

Os sinais do indicador Triple Exponential Moving Average

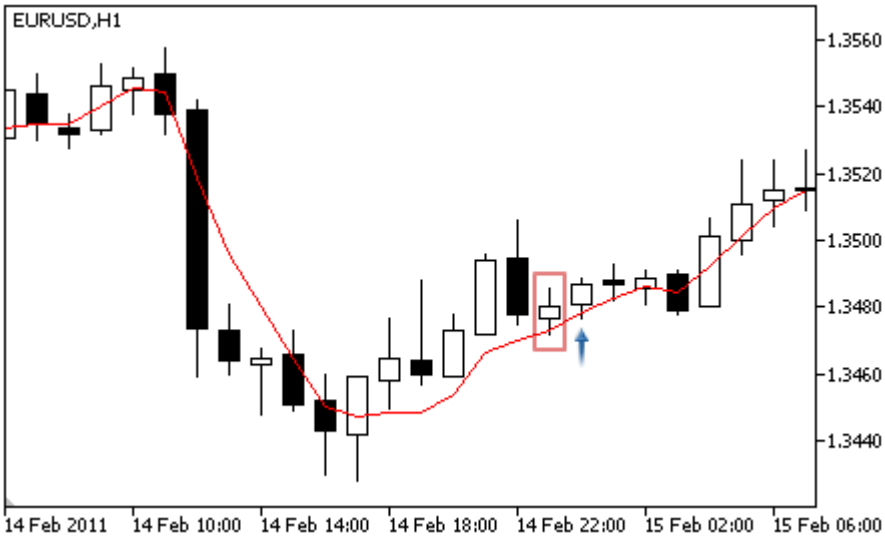
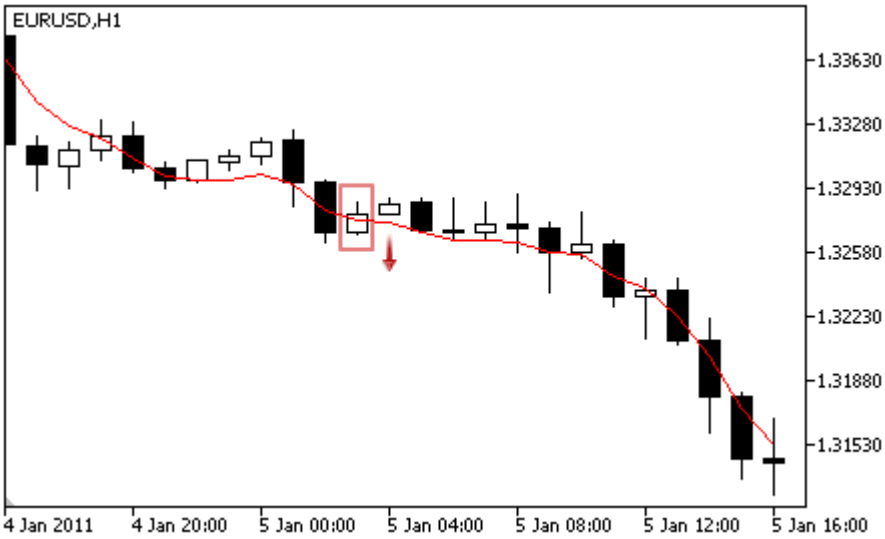
Este módulo de sinais baseia-se nos modelos de mercado do indicador Triplo Exponencial da Média móvel. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

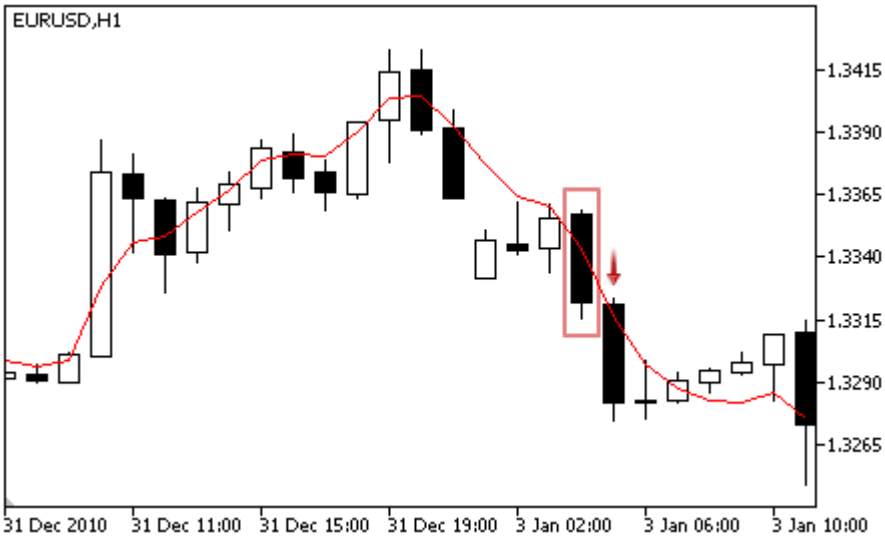
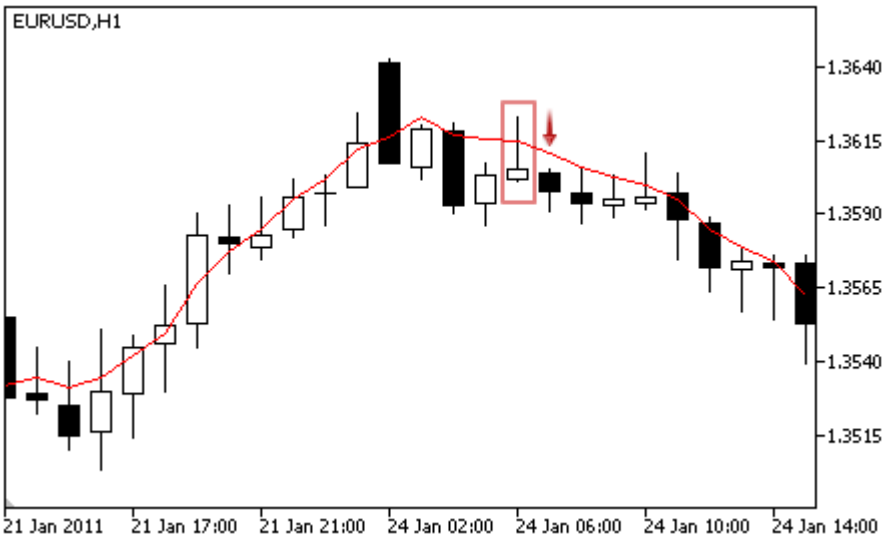
Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para cima (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador aumenta (sinal forte). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	• A sombra inferior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está acima do indicador, e a mínima do preço (Low) está abaixo da linha do indicador) e o indicador aumenta (sinal fraco). 
Para Vender	• O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está abaixo do indicador e o preço de fechamento (Close) está acima da linha do indicador) e o indicador diminui (sinal fraco).  • **Cruzamento Moving Average.** O preço cruza a linha do indicador para baixo (o preço de abertura (Open) da barra analisada está acima do indicador e o preço de fechamento (Close) está abaixo da linha do indicador) e o indicador diminui (sinal forte).

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
	 • A sombra superior da barra cruzou a linha do indicador (os preços de abertura (Open) e fechamento (Close) da barra analisada está abaixo do indicador, e a máxima do preço (High) está acima da linha do indicador) e o indicador diminui (sinal fraco). 
Sem objeções para comprar	Os preços estão acima do indicador.
Sem objeções para vender	O preço está abaixo do indicador.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

© 2000-2024, MetaQuotes Ltd.

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodMA	Período de cálculo da média do indicador.
Shift	Deslocamento do indicador ao longo do eixo de tempo (em barras).
Method	Método de média.
Applied	As séries de preços utilizadas para o cálculo do indicador.

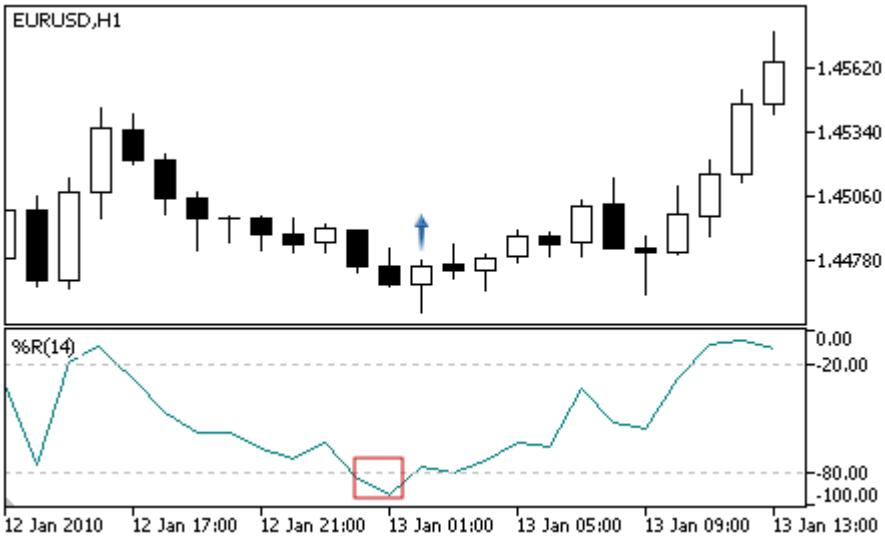
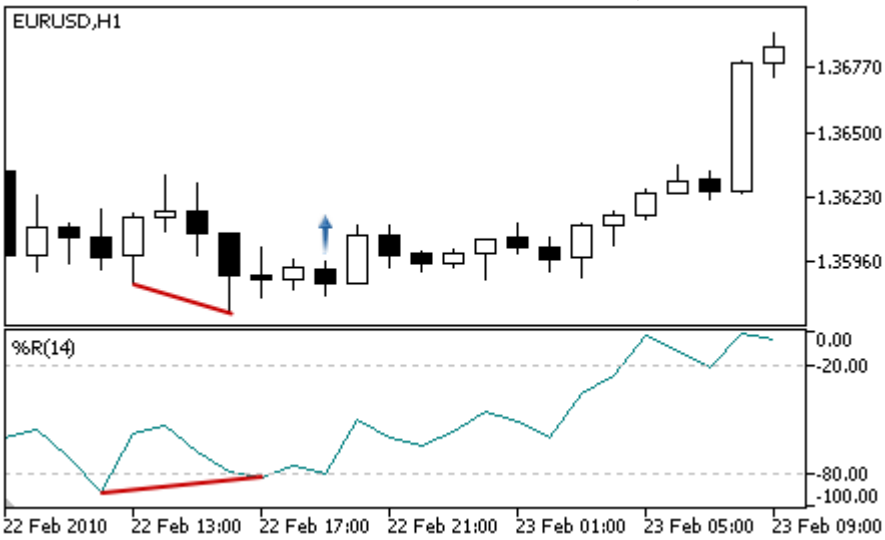
© 2000-2024, MetaQuotes Ltd.

Os sinais do oscilador Williams Percent Range

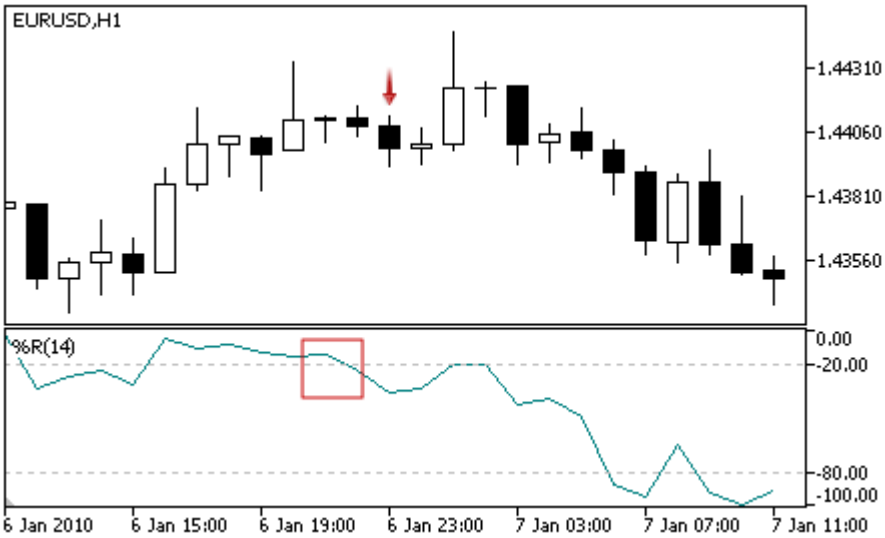
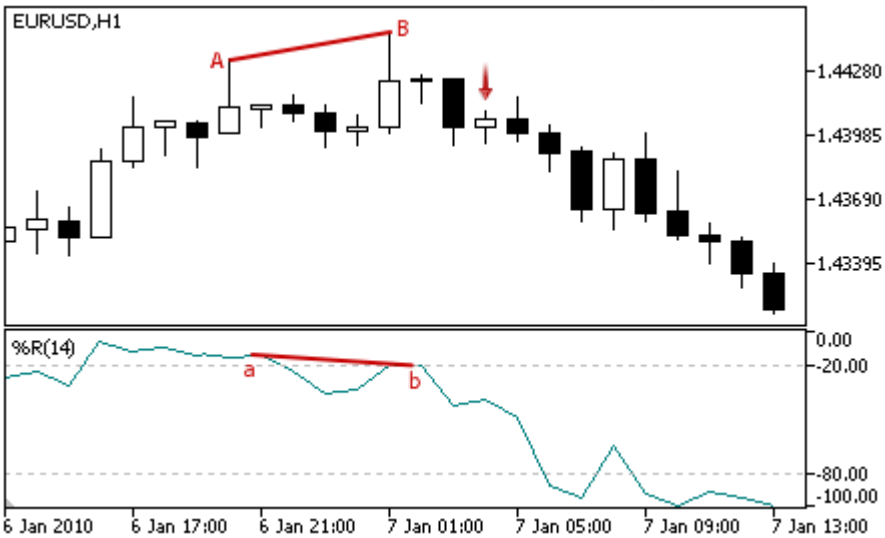
Este módulo de sinais baseia-se nos modelos de mercado do oscilador Williams Percent Range. O mecanismo de tomada de decisões das negociações é baseado em sinais obtidos a partir dos módulos descritos em seção separada.

Condições da Geração dos Sinais

Abaixo temos a descrição das condições do módulo para o sinal a um Expert Advisor.

Tipo de Sinal	Descrição das Condições
Para Comprar	• **Reversão atrás do nível sobrevendido** - O oscilador se volta cima e seu valor na barra analisada está por atrás do nível sobrevendido (valor padrão é de -20).  • **Divergência** - o valor da primeira parte da linha do indicador no fundo (a) é menor do que um ponto posterior (b) em relação ao valor correspondente da mínima (Low) do preço da barra (A) que é maior do que o valor correspondente da mínima (Low) do preço da barra (B). 

© 2000-2024, MetaQuotes Ltd.

Tipo de Sinal	Descrição das Condições
Para Vender	• **Reversão atrás do nível sobrecomprado** - O oscilador se volta para baixo e seu valor na barra analisada está atrás do nível de sobrecomprado (valor padrão é -80).  • **Divergência** - o valor da primeira parte da linha do indicador no topo (a) é maior do que um ponto posterior (b) em relação ao valor correspondente da máxima (High) do preço da barra (A) que é menor do que o valor correspondente da máxima (High) do preço da barra (B). 
Sem objeções para comprar	Valor do oscilador cresce na barra analisada.
Sem objeções para vender	Valor do oscilador decresce na barra analisada.

Observação

Dependendo do modo de operação de um Expert Advisor ("Every tick" ou "Open prices only"), a barra analisada é a barra atual (com índice 0), ou a última barra formada (com índice 1).

© 2000-2024, MetaQuotes Ltd.

Lembre-se que o oscilador Williams Percent Range tem uma escala reversa. O seu valor máximo é -100, o mínimo é 0.

Parâmetros Ajustáveis

Este módulo tem os seguintes parâmetros ajustáveis:

Parâmetro	Descrição
Weight	Peso de sinal do módulo no intervalo de 0 a 1.
PeriodWPR	Período de cálculo do oscilador.

© 2000-2024, MetaQuotes Ltd.

Classes de Trailing Stop

Esta seção contém detalhes técnicos do trabalho com as classes de Trailing Stop (deslocamento do nivel de parada para proteger o capital) e descrição dos componentes importantes da biblioteca padrão MQL5.

O uso destas classes vai economizar tempo ao criar (e testar) estratégias de negociação.

A Biblioteca Padrão MQL5 (em termos de estratégias de negociação) é alocada no diretório do terminal, na pasta Include\Expert\Trailing.

Classe	Descrição
CTrailingFixedPips	Essa classe implementa o algoritmo do Trailing Stop com base em pontos fixos
CTrailingMA	Essa classe implementa o algoritmo do Trailing Stop com base nos valores do indicador Moving Average
CTrailingNone	Uma classe stub, não usa qualquer algoritmo de Trailing Stop
CTrailingPSAR	Essa classe implementa o algoritmo do Trailing Stop com base nos valores do indicador Parabolic SAR

© 2000-2024, MetaQuotes Ltd.

CTrailingFixedPips

CTrailingFixedPips é uma classe que implementação o algoritmo Trailing Stop baseado em pontos fixos para mover a parada das perdas do capital.

Se a posição tem preço Stop Loss, ela verifica a distância Stop Loss mínimo permitido para o preço atual. Se o seu valor for inferior, esse nível Stop Loss sugere definir novo preço Stop Loss. No caso se a posição tem Take Profit, a classe sugere a definição de novo preço Take Profit.

Se o Expert Advisor tiver inicializado com a flag every_tick=false, ele irá realizar todas as operações (de negociação, mover paradas de perdas, etc) apenas na nova barra. Neste caso o nível Take Profit pode ser utilizado. Ele permitirá o fechamento da posição aberta no preço Trailing Stop antes da nova barra ser concluída.

Descrição

CTrailingFixedPips implementa o algoritmo Trailing Stop baseado em pontos fixos para mover a parada da perdas de capital.

Declaração

```
class CTrailingFixedPips: public CExpertTrailing
```

Título

```
#include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

Métodos de classe

Inicialização	
StopLevel	Define o valor do nível de Stop Loss
ProfitLevel	Define o valor do nível de Take Profit
virtual ValidationSettings	Verifica as configurações
Verifica Métodos de Trailing	
virtual CheckTrailingStopLong	Verifica as condições de Trailing Stop da posição comprada
virtual CheckTrailingStopShort	Verifica as condições de Trailing Stop da posição vendida

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2024, MetaQuotes Ltd.

StopLevel

Define o nível de valor Stop Loss (em pontos).

```
void  StopLevel(
   int  stop_level      // Stop Loss level
   )
```

Parâmetros

stop_loss

[in] O valor do nível Stop Loss (em pontos convencionais 2/4-dígitos).

Observação

Se o nível Stop Loss é igual a 0, o Trailing Stop não é usado.

© 2000-2024, MetaQuotes Ltd.

ProfitLevel

Define o valor do nível Take Profit (em pontos).

```
void  ProfitLevel(
   int  profit_level      // Take profit level
   )
```

Parâmetros

profit_level

[in] O valor do nível Take Profit (em pontos convencionais 2/4-dígitos).

Observação

Se o nível de lucro é igual a 0, o trailing stop não é usado.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A função verifica os níveis de Take Profit e Stop Loss. Os valores corretos são 0 e valores maiores do que o stop mínimo para ordens stop ao símbolo (ativo).

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica condições do Trailing Stop da posição comprada.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Se o nível Stop Loss é igual a 0, o Trailing Stop não é usado. Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço de abertura da posição é assumido como um preço base.

Se o preço Bid atual é superior ao preço do nível preço base+stop loss, sugere-se definir novo preço Stop Loss. Neste caso, se a posição já tem nível Take Profit, sugere-se definir novo preço Take Profit igual a preço Bid+nível take profit.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica Trailing condições de parada de posição vendida.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Se o nível Stop Loss é igual a 0, o Trailing Stop não é usado. Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço de abertura da posição é assumido como um preço base.

Se o preço Ask atual é menor do que o nível do preço base-stop loss, sugere-se definir novo preço Stop Loss. Neste caso, se a posição já tem preço Take Profit, sugere-se definir novo preço Take Profit igual a preço Ask-nível take profit.

© 2000-2024, MetaQuotes Ltd.

CTrailingMA

CTrailingMA é uma classe que implementa um algoritmo trailing stop, com base nos valores do indicador da média móvel.

Descrição

CTrailingMA classe implementa um algoritmo trailing stop, com base nos valores do indicador da média móvel da barra anterior (barra completa).

Declaração

```
class CTrailingMA: public CExpertTrailing
```

Título

```
#include <Expert\Trailing\TrailingMA.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Métodos de classe

Inicialização	
Period	Define período da média móvel
Shift	Define o deslocamento da média móvel
Method	Define método de suavização da média móvel
Applied	Define o preço para aplicar na média móvel
virtual InitIndicators	Inicializa indicadores e séries temporais
virtual ValidationSettings	Verifica as configurações
Verifica Métodos de Trailing	
virtual CheckTrailingStopLong	Verifica as condições de Trailing Stop da posição comprada
virtual CheckTrailingStopShort	Verifica as condições de Trailing Stop da posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

Period

Define período da média móvel.

```
void  Period(
   int  period      // Smoothing period
   )
```

Parâmetros

period

[in] Período da média móvel.

© 2000-2024, MetaQuotes Ltd.

Shift

Define o deslocamento da média móvel.

```
void  Shift(
   int  shift      // Shift
   )
```

Parâmetros

shift

[in] Deslocamento da média móvel.

© 2000-2024, MetaQuotes Ltd.

Method

Define o método de suavização da média móvel.

```
void  Method(
   ENUM_MA_METHOD  method      // Smoothing method
   )
```

Parâmetros

method

[in] Método de suavização do indicador média móvel.

© 2000-2024, MetaQuotes Ltd.

Applied

Define preço para aplicar da média móvel.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Applied price
   )
```

Parâmetros

applied

[in] Preço para aplicar da média móvel.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Inicializa indicadores e séries temporais.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators collection pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para os indicadores e coleção de séries temporais (membro da classe CExpert).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A função verifica o período da média móvel, os valores corretos são positivos.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica condições do Trailing Stop da posição comprada.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Primeiro se calcula a máxima permitida do preço Stop Loss, mais próximo do preço atual e calcula o preço Stop Loss usando os valores do indicador média móvel da barra anterior (barra completa).

Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço base é o preço de abertura da posição.

Se o preço do Stop Loss calculado for maior do que o preço base e menor do que o máximo permitido do preço Stop Loss Stop Loss, sugere-se definir novo preço Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica Trailing condições de parada de posição vendida.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Primeiro se calcula a mínima permitida do preço Stop Loss, mais próximo do preço atual e calcula o preço Stop Loss usando os valores do indicador média móvel da barra anterior (barra completa).

Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço base é o preço de abertura da posição.

Se o preço do Stop Loss calculado for maior do que o preço base e menor do que o mínimo permitido do preço Stop Loss, sugere-se definir novo preço Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CTrailingNone

CTrailingNone é uma classe que se deve ter cuidado. Essa classe deve ser usada na inicialização do objeto Trailing se sua estratégia não usa Trailing Stop.

Descrição

A classe CTrailingNone não implementa qualquer algoritmo de Trailing Stop. Os métodos de verificação das condições de Trailing Stop sempre retornam falso.

Declaração

```
class CTrailingNone: public CExpertTrailing
```

Título

```
#include <Expert\Trailing\TrailingNone.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Métodos de classe

Verifica Métodos de Trailing	
virtual CheckTrailingStopLong	Um método tipo mordaça para verificar as condições do Trailing Stop da posição comprada
virtual CheckTrailingStopShort	Um método tipo mordaça para verificar as condições do Trailing Stop da posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica condições do Trailing Stop da posição comprada.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

A função sempre retorna falsa.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica Trailing condições de parada de posição vendida.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // CPositionInfo object pointer
   double&         sl,            // Stop Loss price
   double&         tp             // Take Profit price
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

A função sempre retorna falsa.

© 2000-2024, MetaQuotes Ltd.

CTrailingPSAR

CTrailingPSAR é uma classe que implementa de algoritmo Trailing Stop baseado nos valores do indicador Parabolic SAR.

Descrição

CTrailingPSAR classe implementa de algoritmo Trailing Stop baseado nos valores do indicador Parabolic SAR da barra anterior (barra completa).

Declaração

```
class CTrailingPSAR: public CExpertTrailing
```

Título

```
#include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Métodos de classe

Inicialização	
Step	Define o valor "step" do indicador Parabolic SAR
Maximum	Define o valor "maximum" do indicador Parabolic SAR
virtual InitIndicators	Inicializa indicadores e séries temporais
Verifica Métodos de Trailing	
virtual CheckTrailingStopLong	Verifica as condições de Trailing Stop da posição comprada
virtual CheckTrailingStopShort	Verifica as condições de Trailing Stop da posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Step

Define o valor "Step" do indicador Parabolic SAR.

```
void  Step(
   double  step      // Step
   )
```

Parâmetros

step

[in] O valor "Step" do indicador Parabolic SAR.

© 2000-2024, MetaQuotes Ltd.

Maximum

Define o valor "Maximum" do indicador Parabolic SAR.

```
void  Maximum(
   double  maximum      // Maximum
   )
```

Parâmetros

maximum

[in] O valor "Maximum" do indicador Parabolic SAR.

© 2000-2024, MetaQuotes Ltd.

InitIndicators

Inicializa indicadores e séries temporais.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators collection pointer
   )
```

Parâmetros

indicators

[in] Ponteiro para os indicadores e coleção de séries temporais (membro da classe CExpert).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopLong

Verifica condições do Trailing Stop da posição comprada.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // Pointer
   double&         sl,            // Link
   double&         tp             // Link
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Primeiro se calcula a máxima permitida do preço Stop Loss, mais próximo do preço atual e calcula o preço Stop Loss usando os valores do indicador média móvel da barra anterior (barra completa).

Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço de abertura da posição é assumido como um preço base.

Se o preço do Stop Loss calculado for maior do que o preço base e menor do que o máximo permitido do preço Stop Loss Stop Loss, sugere-se definir novo preço Stop Loss.

© 2000-2024, MetaQuotes Ltd.

CheckTrailingStopShort

Verifica Trailing condições de parada de posição vendida.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,      // Pointer
   double&         sl,            // Link
   double&         tp             // Link
   )
```

Parâmetros

position

[in] Ponteiro do objeto CPositionInfo.

sl

[in][out] Variável de preço para Stop Loss.

tp

[in][out] Variável do preço Take Profit.

Valor de retorno

verdadeiro se as condições são satisfeitas, o contrário é falso.

Observação

Primeiro se calcula a mínima permitida do preço Stop Loss, mais próximo do preço atual e calcula o preço Stop Loss usando os valores do indicador média móvel da barra anterior (barra completa).

Se a posição já tem preço Stop Loss, o seu valor é assumido como um preço base, caso contrário o preço de abertura da posição é assumido como um preço base.

Se o preço do Stop Loss calculado for maior do que o preço base e menor do que o mínimo permitido do preço Stop Loss, sugere-se definir novo preço Stop Loss.

© 2000-2024, MetaQuotes Ltd.

Classes de Gerenciamento de Dinheiro

Esta seção contém detalhes técnicos para trabalhar com classes de dinheiro, gerenciamento de risco e descrição dos componentes importantes da biblioteca padrão MQL5.

O uso destas classes vai economizar tempo ao criar (e testar) estratégias de negociação.

A Biblioteca Padrão MQL5 (em termos de classes de dinheiro e de gestão de riscos) é alocada no diretório do terminal, na pasta Include\Expert\Money\.

Classe	Descrição
CMoneyFixedLot	Esta classe implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com tamanho predefinido de lote fixo.
CMoneyFixedMargin	Esta classe implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com margem fixa predefinida.
CMoneyFixedRisk	Esta classe implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com risco predefinido.
CMoneyNone	Esta classe implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com o tamanho do lote mínimo permitido.
CMoneySizeOptimized	Esta classe implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com o tamanho do lote variável, dependendo dos resultados das ofertas anteriores.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedLot

CMoneyFixedLot é a classe do algoritmo de gerenciamento de dinheiro, baseado em negociação com tamanho predefinido do lote fixo.

Descrição

CMoneyFixedLot implementa o algoritmo de gerenciamento de dinheiro, com base em negociação com tamanho predefinido do lote fixo.

Declaração

```
class CMoneyFixedLot: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Métodos de classe

Inicialização	
Lots	Define o volume de negociação
virtual ValidationSettings	Verifica as configurações
Métodos de Dinheiro e Gerenciamento de Riscos	
virtual CheckOpenLong	Obtém o volume de negociação para posição comprada
virtual CheckOpenShort	Obtém o volume de negociação para posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Lots

Define o volume de negócios (em lotes).

```
void  Lots(
   double  lots      // Lots
   )
```

Parâmetros

lots

[in] O volume de negócios (em lotes).

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Verifica o volume de negociação específico para correção.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função sempre retorna o volume fixo de negociação, definido pelo método Lots.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume de negociação para a posição vendida.

```
virtual double  CheckOpenShort(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para a posição vendida.

Observação

A função sempre retorna o volume fixo de negociação, definido pelo método Lots.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedMargin

CMoneyFixedMargin é o algoritmo da classe de gestão do dinheiro, com base em negociação com margem fixa predefinida.

Descrição

CMoneyFixedMargin implementa o algoritmo do gerenciamento de dinheiro, com base em negociação com margem fixa predefinida.

Declaração

```
class CMoneyFixedMargin: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

Métodos de classe

Métodos de Dinheiro e Gerenciamento de Riscos	
virtual CheckOpenLong	Obtém o volume de negociação para posição comprada
virtual CheckOpenShort	Obtém o volume de negociação para posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função retorna o volume de negociação da posição comprada, usa a margem fixa. A margem é definida pelo parâmetro "Percent" da classe base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume de negociação para posição vendida.

```
virtual double  CheckOpenShort(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para a posição vendida.

Observação

A função retorna o volume de negociação para a posição vendida, usa a margem fixa. A margem é definida pelo parâmetro "Percent" da classe base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CMoneyFixedRisk

CMoneyFixedRisk é uma classe com a implementação do algoritmo de gerenciamento de dinheiro com risco fixo predefinido.

Descrição

A classe CMoneyFixedRisk implementa o algoritmo de gerenciamento de dinheiro com risco fixo predefinido.

Declaração

```
class CMoneyFixedRisk: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

Métodos de classe

Métodos de Dinheiro e Gerenciamento de Riscos	
virtual CheckOpenLong	Obtém o volume de negociação para posição comprada
virtual CheckOpenShort	Obtém o volume de negociação para posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertMoney

Percent, ValidationSettings, CheckReverse

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função do volume de negociação retorna para a posição comprada, usa o risco fixo. O risco é definido pelo parâmetro "Percent" da classe base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume de negociação para posição vendida.

```
virtual double CheckOpenShort(
   double price,      // Price
   double sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para a posição vendida.

Observação

A função retorna o volume de negociação para posição vendida, usa o risco fixo. O risco é definido pelo parâmetro "Percent" da classe base CExpertMoney.

© 2000-2024, MetaQuotes Ltd.

CMoneyNone

CMoneyNone é uma classe que implementa um algoritmo de negociação com o lote mínimo permitido.

Descrição

A classe CMoneyNone implementa a negociação com o lote mínimo permitido.

Declaração

```
class CMoneyNone: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneyNone.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Métodos de classe

Inicialização	
virtual ValidationSettings	Verifica as configurações
Métodos de Dinheiro e Gerenciamento de Riscos	
virtual CheckOpenLong	Obtém o volume de negociação para posição comprada
virtual CheckOpenShort	Obtém o volume de negociação para posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

A função sempre retorna verdadeira.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função sempre retorna o tamanho do lote mínimo.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenShort(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para a posição vendida.

Observação

A função sempre retorna o tamanho do lote mínimo.

© 2000-2024, MetaQuotes Ltd.

CMoneySizeOptimized

CMoneySizeOptimized é uma classe com a implementação do algoritmo de gerenciamento de dinheiro, baseado em negociação com tamanho de lote variável, em função dos resultados das ofertas anteriores.

Descrição

CMoneySizeOptimized implementa o algoritmo de gerenciamento de dinheiro, baseado em negociação com tamanho de lote variável, em função dos resultados das ofertas anteriores

Declaração

```
class CMoneySizeOptimized: public CExpertMoney
```

Título

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Hierarquia de herança

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Métodos de classe

Inicialização	
DecreaseFactor	Define o valor do fator de decréscimo
virtual ValidationSettings	Verifica as configurações
Métodos de Dinheiro e Gerenciamento de Riscos	
virtual CheckOpenLong	Obtém o volume de negociação para posição comprada
virtual CheckOpenShort	Obtém o volume de negociação para posição vendida

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Métodos herdados da classe CExpertMoney

Percent, CheckReverse, CheckClose

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

DecreaseFactor

Define o valor do fator de decréscimo.

```
void  DecreaseFactor(
   double  decrease_factor      // Decrease factor
   )
```

Parâmetros

decrease_factor

[in] Fator de decréscimo.

Observação

O DecreaseFactor define o coeficiente do volume decrescente (em comparação com o volume da posição anterior) para o caso de operações com perdas consecutivas.

© 2000-2024, MetaQuotes Ltd.

ValidationSettings

Verifica as configurações.

```
virtual bool  ValidationSettings()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se o valor do fator de decréscimo é negativo, ele retorna falso. Caso contrário ele retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

CheckOpenLong

Obtém o volume de negociação para posição comprada.

```
virtual double  CheckOpenLong(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função retorna o volume da negociação por uma posição comprada, o volume dependente dos resultados das ofertas anteriores.

© 2000-2024, MetaQuotes Ltd.

CheckOpenShort

Obtém o volume de negociação para a posição vendida.

```
virtual double  CheckOpenShort(
   double  price,      // Price
   double  sl          // Stop Loss price
   )
```

Parâmetros

price

[in] Preço.

sl

[in] preço para finalizar a perda do capital (Stop Loss).

Valor de retorno

O volume de negociação para posição comprada.

Observação

A função retorna o volume da negociação por uma posição vendida, o volume dependente dos resultados das ofertas anteriores.

© 2000-2024, MetaQuotes Ltd.

Classes para Criação Painéis de Controle e Diálogos

Esta seção contém detalhes técnicos de como trabalhar com classes para criação de painéis de controles e descrição dos componentes relevantes da biblioteca padrão MQL5.

O uso destas classes vai economizar tempo ao criar os painéis de controle para programas MQL5 (Expert Advisors e indicadores).

A Biblioteca Padrão MQL5 (em termos de controles) é colocada na pasta de dados do terminal do cliente, no caminho MQL5\Include\Controls.

Exemplos de trabalho com classes podem ser encontrados nos seguintes artigos:

- Como criar um painel gráfico de qualquer complexidade
- Melhoramos o trabalho com Painéis, adicionando transparência, alterando a cor do plano de fundo e herdando da CAppDialog/CWndClient
- Como adicionar rapidamente paneis de controle a indicadores e conselheiros (EA)
- Crie seus próprios painéis gráficos em MQL5
- Criando painéis de controle MQL5 ativos para negociação

O exemplo de Expert Advisor, que ilustra o funcionamento dessas classes podem ser encontrado em MQL5\Expert\Examples\Controls.

Estruturas auxiliares	Descrição
CRect	Estrutura da área retangular
CDateTime	Estrutura para trabalhar com data e tempo

Classes Base	Descrição
CWnd	Classe base para todos os controles
CWndObj	Classe base para controles e caixas de diálogo
CWndContainer	Classe base para controles complexos (contendo controles dependentes)

Controles simples	Descrição
CLabel	Controle, com base no objeto gráfico "Text label"
CBmpButton	Controle, com base no objeto gráfico "label Bitmap"
CButton	Controle, com base no objeto gráfico "Button"
CEdit	Controle, com base no objeto gráfico "Edit field"

© 2000-2024, MetaQuotes Ltd.

Controles simples	Descrição
CPanel	Controle, com base no objeto gráfico "Rectangle label"
CPicture	Controle, com base no objeto gráfico "Bitmap label"

Controles complexos	Descrição
CScroll	Classe base da barra de rolagem
CScrollV	Barra de rolagem vertical
CScrollH	Barra de rolagem horizontal
CWndClient	Classe base da área do cliente com barras de rolagem
CListView	Visualizar lista
CComboBox	Caixa de controle combinado
CCheckBox	Verificar caixa
CCheckGroup	Verificar grupo
CRadioButton	Botão de Radio
CRadioGroup	Grupo de Rádio
CSpinEdit	Editor de rotação
CDialog	Diálogo
CAppDialog	Diálogo de aplicativo

© 2000-2024, MetaQuotes Ltd.

CRect

CRect é uma classe da área retangular do gráfico.

Descrição

CRect é uma classe de área, é definida por coordenadas dos cantos superior esquerdo e inferior direito de um retângulo em coordenadas Cartesianas.

Declaração

```
class CRect
```

Título

```
#include <Controls\Rect.mqh>
```

Métodos de classe

Propriedades	
Left	Obtém/define a coordenada X do canto superior esquerdo
Top	Obtém/define a coordenada Y do canto superior esquerdo
Right	Obtém/define a coordenada X do canto inferior direito
Bottom	Obtém/define a coordenada Y do canto inferior direito
Width	Obtém/Define a largura
Height	Obtém/Define a altura
SetBound	Define novas coordenadas do uso da classe CRect
Move	Define novas coordenadas da classe CRect
Shift	Executa o deslocamento relativo das coordenadas CRect
Contains	Verifica se o ponto está dentro da área da classe CRect
Métodos adicionais	
Format	Obtém a área coordenada como string

© 2000-2024, MetaQuotes Ltd.

Left (Método Get)

Obtém a coordenada X do canto superior esquerdo.

```
int  Left()
```

Valor de retorno

A coordenada X do canto superior esquerdo.

Left (Método Set)

Define a coordenada X do canto superior esquerdo.

```
void  Left(
   const int  x      // new x coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X do canto superior esquerdo.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Top (Método Get)

Obtém a coordenada Y do canto superior esquerdo.

```
int  Top()
```

Valor de retorno

Coordenada Y do canto superior esquerdo.

Top (Método Set)

Define a coordenada Y do canto superior esquerdo.

```
void  Top(
   const int  y      // y coordinate
   )
```

Parâmetros

y

[in] Nova coordenada Y do canto superior esquerdo.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Right (Método Get)

Obtém a coordenada X do canto inferior direito.

```
int  Right()
```

Valor de retorno

Coordenada X do canto inferior direito.

Right (Método Set)

Define a coordenada Y do canto inferior direito.

```
void  Right(
   const int  x      // x coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X do canto inferior direito.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Bottom (Método Get)

Obtém a coordenada Y do canto inferior direito.

```
int  Bottom()
```

Valor de retorno

Coordenada Y do canto inferior direito.

Bottom (Método Set)

Define a coordenada Y do canto inferior direito.

```
void  Bottom(
   const int  y      // y coordinate
   )
```

Parâmetros

y

[in] Nova coordenada Y do canto inferior direito.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Width (Método Get)

Obtém a largura da área.

```
int  Width()
```

Valor de retorno

Largura da área.

Width (Método Set)

Define nova largura da área.

```
virtual bool  Width(
   const int  w      // width
   )
```

Parâmetros

w

[in] Nova largura.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Height (Método Get)

Obtém a altura da área.

```
int  Height()
```

Valor de retorno

Altura da área.

Height (Método Set)

Define nova altura da área.

```
virtual bool  Height(
   const int  h      // height
   )
```

Parâmetros

h

[in] Nova altura.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SetBound

Define novas coordenadas da área, utilizando as coordenadas da classe CRect.

```
void  SetBound(
   const & CRect  rect      // CRect class
   )
```

Valor de retorno

Nenhum.

SetBound

Define novas coordenadas da área.

```
void  SetBound(
   const int  l      // left
   const int  t      // topt
   const int  r      // right
   const int  b      // bottom
   )
```

Parâmetros

l

[in] Coordenada X do canto superior esquerdo.

t

[in] Coordenada Y do canto superior esquerdo.

r

[in] Coordenada X do canto inferior direito.

b

[in] Coordenada Y do canto inferior direito.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Move

Define novas coordenadas da classe CRect.

```
void  Move(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X.

y

[in] Nova coordenada Y.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Shift

Executa o deslocamento relativo das coordenadas da classe CRect.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parâmetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Contains

Verifica se o ponto está dentro da área da classe CRect.

```
bool  Contains(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parâmetros

x

[in] Coordenada X.

y

[in] Coordenada Y.

Valor de retorno

verdadeiro, se o ponto está dentro da área (incluindo as bordas), caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Format

Obtém as coordenadas de área como string.

```
string  Format(
   string &  fmt,     // format
   ) const
```

Parâmetros

fmt

[in] String com formato.

Valor de retorno

String com as coordenadas de área.

© 2000-2024, MetaQuotes Ltd.

CDateTime

CDateTime é uma estrutura para trabalhar com data e tempo.

Descrição

CDateTime é uma estrutura derivada do MqlDateTime, é usado para operação com controles de data e tempo.

Declaração

```
struct CDateTime
```

Título

```
#include <Tools\DateTime.mqh>
```

Métodos de classe

Propriedades	
MonthName	Obtém nome do mês
ShortMonthName	Obtém nome abreviado do mês
DayName	Obtém o nome do dia da semana
ShortDayName	Obtém nome abreviado do dia da semana
DaysInMonth	Obtém o número de dias no mês
Obtém/Define métodos	
DateTime	Obtém/Define a data e o tempo
Date	Define data
Time	Define tempo
Sec	Define segundos
Min	Define minutos
Hour	Define hora
Day	Define dia do mês
Mon	Define mês
Year	Define ano
Métodos adicionais	
SecDec	Subtrai número determinado de segundos
SecInc	Adiciona o número determinado de segundos
MinDec	Subtrai número determinado de minutos

© 2000-2024, MetaQuotes Ltd.

Propriedades	
MinInc	Adiciona o número determinado de minutos
HourDec	Subtrai determinado número de horas
HourInc	Adiciona determinado número de horas
DayDec	Subtrai determinado número de dias
DayInc	Adiciona determinado número de dias
MonDec	Subtrai determinado número de meses
MonInc	Adiciona determinado número de meses
YearDec	Subtrai determinado número de anos
YearInc	Adiciona determinado número de anos

© 2000-2024, MetaQuotes Ltd.

MonthName

Obtém nome do mês.

```
string  MonthName()  const
```

Obtém nome do mês pelo índice.

```
string  MonthName(
   const int      num         // month index
   )  const
```

Parâmetros

num

[in] Índice do mês (1-12).

Valor de retorno

Nome do mês.

© 2000-2024, MetaQuotes Ltd.

ShortMonthName

Obtém nome abreviado do mês

```
string  ShortMonthName()  const
```

Obtém nome abreviado do mês pelo índice.

```
string  ShortMonthName(
   const int      num         // month index
   )  const
```

Parâmetros

num

[in] Índice do mês (1-12).

Valor de retorno

Nome abreviado do mês.

© 2000-2024, MetaQuotes Ltd.

DayName

Obtém o nome do dia da semana.

```
string  DayName()  const
```

Obtém o nome do dia da semana pelo índice.

```
string  DayName(
   const int      num         // day index
   )  const
```

Parâmetros

num

[in] Índice do dia (0-6).

Valor de retorno

Nome do dia.

© 2000-2024, MetaQuotes Ltd.

ShortDayName

Obtém nome abreviado do dia da semana.

```
string  ShortDayName()  const
```

Obtém nome abreviado do dia da semana pelo índice.

```
string  ShortDayName(
   const int      num         // day index
   )  const
```

Parâmetros

num

[in] Índice do dia (0-6).

Valor de retorno

Nome abreviado do dia.

© 2000-2024, MetaQuotes Ltd.

DaysInMonth

Obtém o número de dias no mês.

```
int  DaysInMonth()  const
```

Valor de retorno

Número de dias no mês.

© 2000-2024, MetaQuotes Ltd.

DateTime (Método Get)

Obtém data e tempo.

```
datetime  DateTime()
```

Valor de retorno

Valor do tipo datetime.

DateTime (Método Set)

Define a data e o tempo com o tipo datetime.

```
void  DateTime(
   const datetime      value         // date and time
   )
```

Parâmetros

value

[in] Valor do tipo datetime.

Valor de retorno

Nenhum.

DateTime (Método Set MqlDateTime)

Define a data e o tempo com o tipo MqlDateTime.

```
void  DateTime(
   const MqlDateTime    &value        // date and time
   )
```

Parâmetros

value

[in] Valor do tipo MqlDateTime.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Data (Método Set de data e tempo)

Define a data com o tipo datetime.

```
void  Date(
   const datetime      value         // date
   )
```

Parâmetros

value

[in] Valor do tipo datetime.

Valor de retorno

Nenhum.

Data (Método Set MqlDateTime)

Define a data com o tipo MqlDateTime.

```
void  Date(
   const MqlDateTime   &value        // date
   )
```

Parâmetros

value

[in] Valor do tipo MqlDateTime.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Time (Método Set datetime)

Define tempo com o tipo datetime.

```
void  Time(
   const datetime      value         // time
   )
```

Parâmetros

value

[in] Valor do tipo datetime.

Valor de retorno

Nenhum.

Tempo (Método Set MqlDateTime)

Define tempo com o tipo MqlDateTime.

```
void  Time(
   const MqlDateTime   &value        // time
   )
```

Parâmetros

value

[in] Valor do tipo MqlDateTime.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Sec

Define segundos.

```
void  Sec(
   const int   value        // seconds
   )
```

Parâmetros

value

[in] Segundos.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Min

Define minutos.

```
void  Min(
   const int   value        // minutes
   )
```

Parâmetros

value

[in] Minutos.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Hour

Define hora.

```
void  Hour(
   const int   value        // hour
   )
```

Parâmetros

value

[in] Hora.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Day

Define o dia do mês.

```
void  Day(
   const int   value        // day
   )
```

Parâmetros

value

[In] Dia do mês.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Mon

Define mês.

```
void  Mon(
   const int   value        // month
   )
```

Parâmetros

value

[in] Mês.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Year

Define ano.

```
void  Year(
   const int   value        // year
   )
```

Parâmetros

value

[in] ano.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

SecDec

Subtrai número determinado de segundos.

```
void  SecDec(
   int   delta=1        // seconds
   )
```

Parâmetros

delta

[in] Segundos para subtrair.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

SecInc

Adiciona o número determinado de segundos.

```
void  SecInc(
   int   delta=1        // seconds
   )
```

Parâmetros

delta

[in] Segundos para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MinDec

Subtrai número determinado de minutos.

```
void  MinDec(
   int   delta=1        // minutes
   )
```

Parâmetros

delta

[in] Para subtrair Minutos.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MinInc

Adiciona o número determinado de minutos.

```
void  MinInc(
   int   delta=1        // minutes
   )
```

Parâmetros

delta

[in] Minutos para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

HourDec

Subtrai o número de horas determinado.

```
void  HourDec(
   int   delta=1        // hours
   )
```

Parâmetros

delta

[in] Horas para subtrair.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

HourInc

Adiciona número de horas determinado.

```
void  HourInc(
   int   delta=1        // hours
   )
```

Parâmetros

delta

[in] Horas para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DayDec

Subtrai determinado número de dias.

```
void  DayDec(
   int   delta=1        // days
   )
```

Parâmetros

delta

[in] Dias para subtrair.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

DayInc

Adiciona determinado número de dias.

```
void  DayInc(
   int   delta=1        // days
   )
```

Parâmetros

delta

[in] Dias para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MonDec

Subtrai determinado número de meses.

```
void  MonDec(
   int   delta=1        // months
   )
```

Parâmetros

delta

[in] Meses para subtrair.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MonInc

Adiciona determinado número de meses.

```
void  MonInc(
   int   delta=1        // months
   )
```

Parâmetros

delta

[in] Meses para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

YearDec

Subtrai determinado número de anos.

```
void  YearDec(
   int   delta=1        // years
   )
```

Parâmetros

delta

[in] Anos para subtrair.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

YearInc

Adiciona determinado número de anos.

```
void  YearInc(
   int   delta=1        // years
   )
```

Parâmetros

delta

[in] Anos para adicionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CWnd

CWnd é uma classe base para todos os controles, incluído na biblioteca padrão MQL5.

Descrição

CWnd classe é a implementação da classe de controle base.

Declaração

```
class CWnd : public CObject
```

Título

```
#include <Controls\Wnd.mqh>
```

Hierarquia de herança

CObject
  CWnd

Descendentes diretos

CDragWnd, CWndContainer, CWndObj

Métodos de classe

Criar e destruir	
Create	Criar controle
Destroy	Destruir controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
OnMouseEvent	Manipulador de eventos para o evento CHARTEVENT_MOUSE_MOVE
Name	
Name	Obtém nome do controle
O acesso ao recipiente	
ControlsTotal	Obtém o número de controles no recipiente
Control	Obtém o controle pelo índice
ControlFind	Obtém o controle pelo ID
Geometria	
Rect	Obtém o ponteiro para o objeto da classe CRect

© 2000-2024, MetaQuotes Ltd.

Criar e destruir	
Left	Obtém/Define a coordenada X do canto superior esquerdo
Top	Obtém/Define a coordenada Y do canto superior esquerdo
Right	Obtém/Define a coordenada X do canto inferior direito
Bottom	Obtém/Define a coordenada Y do canto inferior direito
Width	Obtém/Define a largura
Height	Obtém/Define a altura
Move	Define novas coordenadas do controle
Shift	Executa o deslocamento relativo das coordenadas de controle
Resize	Define nova largura/altura do controle
Contains	Verifica se o ponto/controle está dentro da zona de controle
Align	
Alignment	Define as propriedades de alinhamento do controle
Align	Executa o controle de alinhamento
Identificação	
Id	Obtém/Define a ID de controle
State	
IsEnabled	Obtém um valor indicando se o controle está habilitado
Enable	Define um valor indicando se o controle está habilitado
Desabilitar	Desabilita o controle
IsVisible	Verifica flag de visibilidade
Visible	Define flag de visibilidade
Show	Mostra o controle
Hide	Oculta o controle
IsActive	Verifica a atividade de controle
Activate	Ativa o controle
Deactivate	Desativa o controle
Flags de Estado	
StateFlags	Obtém/Define flags de estado de controle
StateFlagsSet	Define flags do estado de controle
StateFlagsReset	Redefine flags do estado de controle

© 2000-2024, MetaQuotes Ltd.

Criar e destruir	
Flags das propriedades	
PropFlags	Obtém/Define os flags das propriedades de controle
PropFlagsSet	Define os flags das propriedades de controle
PropFlagsReset	Redefine os flags das propriedades de controle
Operações de mouse	
MouseX	Obtém/Salva coordenada X do mouse
MouseY	Obtém/Salva a coordenada Y do mouse
MouseFlags	Obtém/Salva o estado dos botões do mouse
MouseFocusKill	Limpa o foco do mouse
Manipuladores de eventos internos	
OnCreate	Manipulador de evento "Create"
OnDestroy	Manipulador de evento "Destroy"
OnMove	Manipulador de evento "Move"
OnResize	Manipulador de evento "Resize"
OnEnable	Manipulador de evento "Enable"
OnDisable	Manipulador de evento "Disable"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnActivate	Manipulador de evento "Activate"
OnDeactivate	Manipulador de evento "Deactivate"
OnClick	Manipulador de eventos "Clique"
OnChange	Manipulador de eventos "Change"
Manipuladores de eventos do mouse	
OnMouseDown	Manipulador de eventos "MouseDown"
OnMouseUp	Manipulador de eventos "MouseUp"
Manipulador de eventos para arrastar	
OnDragStart	Manipulador de eventos "DragStart"
OnDragProcess	Manipulador de eventos "DragProcess"

© 2000-2024, MetaQuotes Ltd.

Criar e destruir	
OnDragEnd	Manipulador de eventos "Dragend"
Arrastar o objeto	
DragObjectCreate	Cria objeto de arrastar
DragObjectDestroy	Destrói objeto de arrastar

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2024, MetaQuotes Ltd.

Create

Cria um controle.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Método de classe base somente salva os parâmetros e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

Destroy

Destrói um controle.

```
virtual bool  Destroy()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro de evento do tipo long, passado por referência.

dparam

[in] Parâmetro de evento do tipo double, passado por referência.

sparam

[in] Parâmetro de evento do tipostring, passado por referência.

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMouseEvent

Manipulador de eventos do mouse (o evento gráfico CHARTEVENT_MOUSE_MOVE).

```
virtual bool  OnMouseEvent(
   const int  x,          // x coordinate
   const int  y,          // y coordinate
   const int  flags       // flags
   )
```

Parâmetros

x

[in] Coordenada X do cursor do mouse em relação ao canto superior esquerdo do gráfico.

y

[in] Coordenada Y do cursor do mouse em relação ao canto superior esquerdo do gráfico.

flags

[in] Flag de estado dos botões do mouse.

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Name

Obtém o controle do nome.

```
string Name() const
```

Valor de retorno

Nome do controle.

© 2000-2024, MetaQuotes Ltd.

ControlsTotal

Obtém o número de controles no recipiente.

```
int ControlsTotal() const
```

Valor de retorno

Número de controles no recipiente.

Observação

O método da classe base não tem o recipiente, ele fornece o acesso ao recipiente a seus herdeiros e sempre retorna 0.

© 2000-2024, MetaQuotes Ltd.

Control

Obtém o controle pelo índice.

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Parâmetros

ind

[in] índice de controle.

Valor de retorno

Um ponteiro para o controle.

Observação

O método da classe base não tem o recipiente, ele fornece o acesso ao recipiente a seus herdeiros e sempre retorna NULL.

© 2000-2024, MetaQuotes Ltd.

ControlFind

Obtém o controle de recipiente pela ID determinada.

```
virtual CWnd*  ControlFind(
   const long  id      // ID
   )
```

Parâmetros

id

[in] Identificador do controle para encontrar.

Valor de retorno

Ponteiro do controle.

Observação

O método da classe base não tem o recipiente, ele fornece o acesso ao recipiente a seus herdeiros. Se a ID determinada corresponde com a ID de recipiente, retorna um ponteiro para si.

© 2000-2024, MetaQuotes Ltd.

Rect

Obtém o ponteiro ao objeto da classe CRect.

```
const CRect*  Rect()  const
```

Valor de retorno

Ponteiro ao objeto da classe CRect.

© 2000-2024, MetaQuotes Ltd.

Left (Método Get)

Obtém a coordenada X do canto superior esquerdo do controle.

```
int  Left()
```

Valor de retorno

Coordenada X do canto superior esquerdo do controle.

Left (Método Set)

Define a coordenada X do canto superior esquerdo do controle.

```
void  Left(
   const int  x      // new x coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X do canto superior esquerdo.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Top (Método Get)

Obtém a coordenada Y do canto superior esquerdo do controle.

```
int  Top()
```

Valor de retorno

Coordenada Y do canto superior esquerdo do controle.

Top (Método Set)

Define a coordenada Y do canto superior esquerdo do controle.

```
void  Top(
   const int  y      // y coordinate
   )
```

Parâmetros

y

[in] Nova coordenada Y do canto superior esquerdo.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Right (Método Get)

Obtém a coordenada X do canto inferior direito do controle.

```
int  Right()
```

Valor de retorno

Coordenada X do canto inferior direito.

Right (Método Set)

Define a coordenada X do canto inferior direito do controle.

```
void  Right(
   const int  x      // x coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X do canto inferior direito.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Bottom (Método Get)

Obtém a coordenada Y do canto inferior direito do controle.

```
int  Bottom()
```

Valor de retorno

Coordenada Y do canto inferior direito do controle.

Bottom (Método Set)

Define a coordenada X do canto inferior direito do controle.

```
void  Bottom(
   const int  y      // y coordinate
   )
```

Parâmetros

y

[in] Nova coordenada Y do canto inferior direito.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Width (Método Get)

Obtém a largura do controle.

```
int  Width()
```

Valor de retorno

Width of the control.

Width (Método Set)

Define nova largura do controle.

```
virtual bool  Width(
   const int  w      // width
   )
```

Parâmetros

w

[in] Nova largura.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Height (Método Get)

Obtém a altura do controle.

```
int  Height()
```

Valor de retorno

Altura do controle.

Height (Método Set)

Define nova altura do controle.

```
virtual bool  Height(
   const int  h      // height
   )
```

Parâmetros

h

[in] Nova altura.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Move

Define novas coordenadas do controle.

```
void  Move(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X.

y

[in] Nova coordenada Y.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Shift

Executa a deslocamento relativo das coordenadas de controle.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parâmetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Resize

Define nova largura/altura do controle.

```
virtual bool  Resize(
   const int  w,      // width
   const int  h       // height
   )
```

Parâmetros

w

[in] Nova largura.

h

[in] Nova altura.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Contains

Verifica se o ponto está dentro da área de controle do gráfico.

```
bool  Contains(
   const int  x,      // X coordinate
   const int  y       // Y coordinate
   )
```

Parâmetros

x

[in] Coordenada X.

y

[in] Coordenada Y.

Valor de retorno

verdadeiro, se o ponto está dentro da área (incluindo as bordas), caso contrário - falso.

Contains

Verifica se o controle determinado está dentro da área de controle do gráfico.

```
bool  Contains(
   const CWnd*  control      // pointer
   )  const
```

Parâmetros

control

[in] Ponteiro de objeto.

Valor de retorno

verdadeiro, se o controle determinado está dentro da área (incluindo as bordas), caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Alignment

Define os parâmetros de alinhamento do controle.

```
void  Alignment(
   const int  flags,       // alignment flags
   const int  left,        // offset from the left border
   const int  top,         // offset from the top border
   const int  right,       // offset from the right border
   const int  bottom       // offset from the bottom border
   )
```

Parâmetros

flags

[in] Flags de alinhamento.

left

[in] Deslocamento fixo a partir da borda esquerda.

top

[in] Deslocamento fixo a partir da borda superior.

right

[in] Deslocamento fixo a partir da borda direita.

bottom

[in] Deslocamento fixo a partir da borda inferior.

Valor de retorno

Nenhum.

Observação

Flags de alinhamento:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // no align
   WND_ALIGN_LEFT=1,                                  // align left
   WND_ALIGN_TOP=2,                                   // align top
   WND_ALIGN_RIGHT=4,                                 // align right
   WND_ALIGN_BOTTOM=8,                                // align bottom
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // align width
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // align height
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // align height and width
   }
```

© 2000-2024, MetaQuotes Ltd.

Align

Executa o alinhamento do controle na área do gráfico determinado.

```
virtual bool  Align(
   const CRect*  rect      // pointer
   )
```

Parâmetros

rect

[in] Ponteiro ao objeto com coordenadas da área do gráfico.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Os parâmetros de alinhamento deve ser especificados (sem alinhamento por padrão).

© 2000-2024, MetaQuotes Ltd.

Id (Método Get)

Obtém a ID de controle.

```
long  Id()  const
```

Valor de retorno

O identificador de controle.

Id (Método Set)

Define novo valor do ID de controle.

```
virtual long  Id(
   const long  id      // identifier
   )
```

Parâmetros

id

[in] O novo valor do identificador de controle.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

IsEnabled

Obtém um valor indicando se o controle está habilitado.

```
bool IsEnabled() const
```

Valor de retorno

verdadeiro - se o controle está habilitado, caso contrário - falso.

© 2000-2024, MetaQuotes Ltd.

Enable

Permite o controle.

```
virtual bool  Enable()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

Se o controle está habilitado, poderá processar os eventos externos.

© 2000-2024, MetaQuotes Ltd.

Desabilitar

Desabilita o controle.

```
virtual bool  Disable()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

O controle desabilitado não poderá processar os eventos externos.

© 2000-2024, MetaQuotes Ltd.

IsVisible

Obtém um valor indicando se o controle está visível.

```
bool IsVisible() const
```

Valor de retorno

verdadeiro se o controle é mostrado no gráfico, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

Visible

Define flag de visibilidade.

```
virtual bool  Visible(
   const bool  flag      // flag
   )
```

Parâmetros

flag

[in] Novo flag

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Show

Exibe o controle.

```
virtual bool  Show()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Hide

Oculta o controle.

```
virtual bool  Hide()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

IsActive

Obtém um valor indicando se o controle está ativo.

```
bool IsActive() const
```

Valor de retorno

verdadeiro se o controle está ativo, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

Activate

Ativa o controle.

```
virtual bool  Activate()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

O controle se torna ativo quando o cursor do mouse está pairando sobre ele.

© 2000-2024, MetaQuotes Ltd.

Deactivate

Desativa o controle.

```
virtual bool  Deactivate()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

O controle torna-se inativo quando o cursor do mouse está fora de controle.

© 2000-2024, MetaQuotes Ltd.

StateFlags (Método Get)

Obtém flags de estado do controle.

```
int  StateFlags()
```

Valor de retorno

Flags de estado do controle.

StateFlags (Método Set)

Define flags de estado do controle.

```
virtual void  StateFlags(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Novos flags de estado do controle.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

StateFlagsSet

Define flags de estado do controle.

```
virtual void  StateFlagsSet(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags para definir (máscara de bits).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

StateFlagsReset

Redefine flags de estado do controle.

```
virtual void  StateFlagsReset(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags para redefinir (máscara de bits).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

PropFlags (Método Get)

Obtém flags de propriedades do controle.

```
void  PropFlags(
   const int  flags      // flags
   )
```

Valor de retorno

Flags de propriedades do controle.

PropFlags (Método Set)

Define flags de propriedades do controle.

```
virtual void  PropFlags(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Novos flags.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

PropFlagsSet

Define flags de propriedades do controle.

```
virtual void  PropFlagsSet(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags para definir (máscara de bits).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

PropFlagsReset

Redefine flags das propriedade do controle.

```
virtual void  PropFlagsReset(
   const int  flags      // flags
   )
```

Parâmetros

flags

[in] Flags para redefinir (máscara de bits).

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MouseX (Método Set)

Salva a coordenada X do mouse.

```
void  MouseX(
   const int  value      // coordinate
   )
```

Parâmetros

value

[in] A coordenada X do mouse.

Valor de retorno

Nenhum.

MouseX (Método Get)

Obtém a coordenada X do mouse que foi salva.

```
int  MouseX()
```

Valor de retorno

A coordenada X do mouse que foi salva.

© 2000-2024, MetaQuotes Ltd.

MouseY (Método Set)

Salva a coordenada Y do mouse.

```
void  MouseY(
   const int  value      // coordinate
   )
```

Parâmetros

value

[in] A coordenada Y do mouse.

Valor de retorno

Nenhum.

MouseY (Método Get)

Obtém a coordenada Y do mouse que foi salva.

```
int  MouseY()
```

Valor de retorno

A coordenada Y do mouse que foi salva.

© 2000-2024, MetaQuotes Ltd.

MouseFlags (Método Set)

Salva o estado dos botões do mouse.

```
virtual void  MouseFlags(
   const int  value      // state
   )
```

Parâmetros

value

[in] Estado dos botões do mouse.

Valor de retorno

Nenhum.

MouseFlags (Método Get)

Obtém o estado salvo dos botões do mouse.

```
int  MouseFlags()
```

Valor de retorno

Estado dos botões do mouse.

© 2000-2024, MetaQuotes Ltd.

MouseFocusKill

Limpa o estado salvo dos botões do mouse e desativa o controle.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parâmetros

id=CONTROLS_INVALID_ID

[in] Identificador do controle, que recebeu o foco do mouse.

Valor de retorno

O resultado da desativação do controle.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnDestroy

O manipulador de eventos do controle "Destroy".

```
virtual bool  OnDestroy()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnEnable

O manipulador de eventos do controle "Enable" (se habilitado, ele pode responder a interação do usuário).

```
virtual bool  OnEnable()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnDisable

O manipulador de eventos do controle "Disable" (se desativado não pode responder a interação do usuário).

```
virtual bool  OnDisable()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnActivate

O manipulador de eventos do controle "Activate".

```
virtual bool  OnActivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

O manipulador de eventos do controle "Deactivate" .

```
virtual bool  OnDeactivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnClick

O manipulador de eventos do controle "Click" (clique do botão esquerdo do mouse) .

```
virtual bool  OnClick()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

O manipulador de eventos do controle "MouseDown".

```
virtual bool  OnMouseDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseDown" ocorre quando o botão esquerdo do mouse é pressionado no controle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

O manipulador de eventos do controle "MouseUp" (botão esquerdo do mouse liberando o botão).

```
virtual bool  OnMouseUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseUp" ocorre quando o botão esquerdo do mouse é liberado no controle.

© 2000-2024, MetaQuotes Ltd.

OnDragStart

O manipulador de eventos do controle "DragStart".

```
virtual bool  OnDragStart()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "DragStart" ocorre no início da operação de arrastar.

© 2000-2024, MetaQuotes Ltd.

OnDragProcess

O manipulador de eventos do controle "DragProcess" .

```
virtual bool  OnDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parâmetros

x

[in] Coordenada X atual do cursor do mouse.

y

[in] Coordenada Y atual do cursor do mouse.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "DragProcess" ocorre quando o controle é movido.

© 2000-2024, MetaQuotes Ltd.

OnDragEnd

O manipulador de eventos do controle "dragend" .

```
virtual bool  OnDragEnd()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "dragend" ocorre quando o processo de arrastar o controle é finalizado.

© 2000-2024, MetaQuotes Ltd.

DragObjectCreate

Cria objeto de arrastar .

```
virtual bool  DragObjectCreate()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

DragObjectDestroy

Destrói objeto de arrastar.

```
virtual bool  DragObjectDestroy()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CWndObj

CWndObj é uma classe base para controles simples (baseado nos objetos de gráfico) da Biblioteca Padrão.

Descrição

A classe CWndObj implementa métodos de base para controle simples.

Declaração

```
class CWndObj : public CWnd
```

Título

```
#include <Controls\WndObj.mqh>
```

Hierarquia de herança

COBJECT
  CWnd
    CWndObj

Descendentes diretos

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Métodos de classe

Processamento de eventos de gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
Text	Obtém/Define propriedade do objeto de gráfico OBJPROP_TEXT
Color	Obtém/Define propriedade do objeto de gráfico OBJPROP_COLOR
ColorBackground	Obtém/Define propriedade do objeto de gráfico OBJPROP_BGCOLOR
ColorBorder	Obtém/Define propriedade do objeto de gráfico OBJPROP_BORDER_COLOR
Font	Obtém/Define propriedade do objeto de gráfico OBJPROP_FONT
FontSize	Obtém/Define propriedade do objeto de gráfico OBJPROP_FONTSIZE
ZOrder	Obtém/Define propriedade do objeto de gráfico OBJPROP_ZORDER

© 2000-2024, MetaQuotes Ltd.

Processamento de eventos de gráfico	
Manipuladores de eventos dos objetos de gráfico	
OnObjectCreate	Manipulador de Evento CHARTEVENT_OBJECT_CREATE
OnObjectChange	Manipulador de evento CHARTEVENT_OBJECT_CHANGE
OnObjectDelete	Manipulador de evento CHARTEVENT_OBJECT_DELETE
OnObjectDrag	Manipulador de evento CHARTEVENT_OBJECT_DRAG
Propriedades que alteram manipuladores de eventos	
OnSetText	Manipulador de eventos "SetText"
OnSetColor	Manipulador de eventos "SetColor"
OnSetColorBackground	Manipulador de eventos "SetColorBackground"
OnSetFont	Manipulador de eventos "SetFont"
OnSetFontSize	Manipulador de eventos "SetFontSize"
OnSetZOrder	Manipulador de evento "SetZOrder"
Manipuladores de eventos internos	
OnDestroy	Manipulador de evento "Destroy"
OnChange	Manipulador de eventos "Change"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Text (Método Get)

Obtém a propriedade do objeto de gráfico OBJPROP_TEXT(texto).

```
string  Text()
```

Valor de retorno

O valor da propriedade OBJPROP_TEXT.

Text (Método Set)

Define a propriedade do objeto de gráfico OBJPROP_TEXT(texto).

```
bool  Text(
   const string  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade OBJPROP_TEXT.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Color (Método Get)

Obtém a propriedade OBJPROP_COLOR (cor) do objeto de gráfico.

```
color  Color()
```

Valor de retorno

O valor da propriedade OBJPROP_COLOR.

Color (Método Set)

Define a propriedade OBJPROP_COLOR (cor) do objeto de gráfico.

```
bool  Color(
   const color  value      // value
   )
```

Parâmetros

value

[in] Novo valor da propriedade OBJPROP_COLOR

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ColorBackground (Método Get)

Obtém a propriedade OBJPROP_BGCOLOR (cor de fundo) do objeto gráfico.

```
color  ColorBackground()
```

Valor de retorno

O valor da propriedade OBJPROP_BGCOLOR.

ColorBackground (Método Set)

Define a propriedade OBJPROP_BGCOLOR (cor de fundo) do objeto gráfico.

```
bool  ColorBackground(
   const color  value      // value
   )
```

Parâmetros

value

[in] Novo valor da propriedade OBJPROP_BGCOLOR.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ColorBorder (Método Get)

Obtém a propriedade OBJPROP_BORDER_COLOR (cor da borda) do objeto de gráfico.

```
color  ColorBorder()
```

Valor de retorno

O valor da propriedade OBJPROP_BORDER_COLOR.

ColorBorder (Método Set)

Define a propriedade OBJPROP_BORDER_COLOR (cor da borda) do objeto de gráfico.

```
bool  ColorBorder(
   const color  value      // value
   )
```

Parâmetros

value

[in] Novo valor da propriedade OBJPROP_BORDER_COLOR.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Font (Método Get)

Obtém a propriedade OBJPROP_FONT (fonte) do objeto de gráfico.

```
string  Font()
```

Valor de retorno

O valor da propriedade OBJPROP_FONT.

Font (Método Set)

Define a propriedade OBJPROP_FONT (fonte) do objeto de gráfico.

```
bool  Font(
   const string  value      // new value
   )
```

Parâmetros

value

[in] Novo valor da propriedade OBJPROP_FONT.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

FontSize (Método Get)

Obtém a propriedade< OBJPROP_FONTSIZE (tamanho da fonte) do objeto de gráfico.

```
int  FontSize()
```

Valor de retorno

O valor da propriedade< OBJPROP_FONTSIZE.

FontSize (Método Set)

Define a propriedade< OBJPROP_FONTSIZE (tamanho da fonte) do objeto de gráfico.

```
bool  FontSize(
   const int  value      // new value
   )
```

Parâmetros

value

[in] Novo valor da propriedade< OBJPROP_FONTSIZE.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ZOrder (Método Get)

Obtém a propriedade do objeto de gráfico OBJPROP_ZORDER.

```
long  ZOrder()
```

Valor de retorno

O valor da propriedade OBJPROP_ZORDER.

ZOrder (Método Set)

Define a propriedade do objeto de gráfico OBJPROP_ZORDER.

```
bool  ZOrder(
   const long  value      // new value
   )
```

Parâmetros

value

[in] Novo valor da propriedade OBJPROP_ZORDER.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnObjectCreate

O manipulador de eventos do objeto de gráfico CHARTEVENT_OBJECT_CREATE.

```
virtual bool  OnObjectCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnObjectChange

O manipulador de eventos do objeto de gráfico CHARTEVENT_OBJECT_CHANGE.

```
virtual bool  OnObjectChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnObjectDelete

O manipulador de eventos do objeto de gráfico CHARTEVENT_OBJECT_DELETE.

```
virtual bool  OnObjectDelete()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnObjectDrag

O manipulador de eventos do objeto de gráfico CHARTEVENT_OBJECT_DRAG.

```
virtual bool  OnObjectDrag()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetText

O manipulador de eventos do controle "SetText" (altera propriedade OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

O manipulador de eventos do controle "SetColor" (altera propriedade OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

O manipulador de eventos do controle "SetColorBackground" (altera propriedade OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

O manipulador de eventos do controle "SetFont" (altera propriedade OBJPROP_FONT).

```
virtual bool  OnSetFont()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

O manipulador de eventos controle "SetFontSize" (altera propriedade OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

O manipulador de eventos do controle "SetZOrder" (altera propriedade OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnDestroy

O manipulador de eventos do controle "Destroy".

```
virtual bool  OnDestroy()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CWndContainer

CWndContainer é uma classe base para um controle complexo (contendo controles dependentes) da biblioteca padrão.

Descrição

A classe CWndContainer implementa métodos de base do controle complexo.

Declaração

```
class CWndContainer : public CWnd
```

Título

```
#include <Controls\WndContainer.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer

Descendentes diretos

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Métodos de classe

Destroy	
Destroy	Destrói todos os controles de recepiente
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
OnMouseEvent	O manipulador de evento CHARTEVENT_MOUSE_MOVE
O acesso ao recipiente	
ControlsTotal	Obtém o Número dos controles no recipiente
Control	Obtém o controle pelo índice
ControlFind	Obtém o controle pela ID
Adicionar/Deletar	
Add	Adiciona controle para recipiente
Delete	Exclui controle de recipiente
Geometria	

© 2000-2024, MetaQuotes Ltd.

Destroy	
Move	Define novas coordenadas para todos os controles do recipiente
Shift	Executa o deslocamento relativo das coordenadas para todos os controles do recipiente
Identificação	
Id	Define a ID para todos os controles do recipiente
State	
Enable	Permite todos os controles do recipiente
Desabilitar	Desativa todos os controles do recipiente
Show	Mostra todos os controles do recipiente
Hide	Oculta todos os controles do recipiente
Operações de mouse	
MouseFocusKill	Limpa o foco do mouse
Operações de arquivo	
Save	Salva as informações do recipiente para o arquivo
Load	Carrega as informações do recipiente a partir do arquivo
Manipuladores de eventos internos	
OnResize	Manipulador de evento "Resize"
OnActivate	Manipulador de evento "Activate"
OnDeactivate	Manipulador de evento "Deactivate"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

Destroy

Destrói todos os controles do recipiente.

```
virtual bool  Destroy()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMouseEvent

Manipulador do evento Mouse.

```
virtual bool  OnMouseEvent(
   const int  x,          // x coordinate
   const int  y,          // y coordinate
   const int  flags       // flags
   )
```

Parâmetros

x

[in] Coordenada X do cursor do mouse em relação ao canto superior esquerdo do gráfico.

y

[in] Coordenada Y do cursor do mouse em relação ao canto superior esquerdo do gráfico.

flags

[in] Flag de estado dos botões do mouse.

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

ControlsTotal

Obtém o número de controles no recipiente.

```
int ControlsTotal() const
```

Valor de retorno

Número de controles no recipiente.

© 2000-2024, MetaQuotes Ltd.

Control

Obtém o controle do recipiente pelo índice.

```
CWnd*  Control(
   const int  ind      // index
   )  const
```

Parâmetros

ind

[in] O índice necessário do controle.

Valor de retorno

Ponteiro para o controle, caso contrário é NULL se o controle não for encontrado.

© 2000-2024, MetaQuotes Ltd.

ControlFind

Obtém o controle do recipiente pelo identificador.

```
virtual CWnd*  ControlFind(
   const long  id      // id
   )
```

Parâmetros

id

[in] ID de Controle .

Valor de retorno

Ponteiro para o controle, caso contrário é NULL se o controle não for encontrado.

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona um controle ao recipiente.

```
bool  Add(
   CWnd&  control      // reference
   )
```

Parâmetros

control

[in] Controle para adicionar, passado por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Delete

Exclui controle a partir do recipiente.

```
bool  Delete(
   CWnd&  control      // reference
   )
```

Parâmetros

control

[in] Controle para excluir, passado por referência.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Move

Define novas coordenadas para todos os controles do recipiente.

```
virtual bool  Move(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parâmetros

x

[in] Nova coordenada X do canto superior esquerdo.

y

[in] Nova coordenada Y do canto superior esquerdo.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Shift

Executa o deslocamento relativo das coordenadas a todos os controles do recipiente.

```
virtual bool  Shift(
   const int  dx,      // delta x
   const int  dy       // delta y
   )
```

Parâmetros

dx

[in] Delta X.

dy

[in] Delta Y.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Id

Define a ID para todos os controles do recipiente.

```
virtual long  Id(
   const long  id      // identifier
   )
```

Parâmetros

id

[in] Identificador do grupo Base.

Valor de retorno

Número de identificadores, usado pelos controles de recipiente.

© 2000-2024, MetaQuotes Ltd.

Enable

Permite todos os controles do recipiente.

```
virtual bool  Enable()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Desabilitar

Desativa todos os controles do recipiente.

```
virtual bool  Disable()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Show

Mostra todos os controles do recipiente.

```
virtual bool  Show()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Hide

Oculta todos os controles do recipiente.

```
virtual bool  Hide()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

MouseFocusKill

Limpa o estado de salvo dos botões do mouse e desativa todos os controles no recipiente.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parâmetros

id=CONTROLS_INVALID_ID

[in] Identificador do controle, que recebeu o foco do mouse.

Valor de retorno

O resultado de desativação dos controles.

© 2000-2024, MetaQuotes Ltd.

Save

Salva informações do recepiente no arquivo.

```
virtual bool  Save(
   const int  file_handle      // handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário (deve ser aberto para gravação).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Load

Carrega as informações do recepiente no arquivo

```
virtual bool  Load(
   const int  file_handle      // handle
   )
```

Parâmetros

file_handle

[in] Manipulador do arquivo binário (deve ser aberto para leitura).

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnActivate

O manipulador de eventos do controle "Activate".

```
virtual bool  OnActivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

O manipulador de eventos do controle "Deactivate" .

```
virtual bool  OnDeactivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

CLabel

CLabel é uma classe de controle simples, baseado no objeto de gráfico "Text label".

Descrição

CLabel destina-se a criação de etiquetas simples de texto.

Declaração

```
class CLabel : public CWndObj
```

Título

```
#include <Controls\Label.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CLabel

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriedades que alteram manipuladores de eventos	
OnSetText	Manipulador de eventos "SetText"
OnSetColor	Manipulador de eventos "SetColor"
OnSetFont	Manipulador de eventos "SetFont"
OnSetFontSize	Manipulador de eventos "SetFontSize"
Manipuladores de eventos internos	
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com etiqueta bitmap:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
```

```
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                        // CLabel object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- create dependent controls
   bool              CreateLabel(void);
   //--- handlers of the dependent controls events
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateLabel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CLabel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- coordinates
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- create
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CLabel.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnSetText

O manipulador de eventos do controle "SetText" (altera a propriedade OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

O manipulador de eventos do controle "SetColor" (altera a propriedade OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

O manipulador de eventos do controle "SetFont" (altera propriedade OBJPROP_FONT.

```
virtual bool  OnSetFont()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

O manipulador de eventos do controle "SetFontSize" (altera a propriedade OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CBmpButton

CBmpButton é uma classe de controle simples, baseado no objeto gráfico "Bitmap label".

Descrição

CBmpButton destina-se a criação de botões com imagem gráfica.

Declaração

```
class CBmpButton : public CWndObj
```

Título

```
#include <Controls\BmpButton.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CBmpButton

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriedades	
Border	Obtém/Define a propriedade de controle "border"
BmpNames	Define o nome do arquivo bmp de controle
BmpOffName	Obtém/Define o nome do arquivo bmp para o estado OFF
BmpOnName	Obtém/Define o nome do arquivo bmp para o estado ON
BmpPassiveName	Obtém/Define o nome do arquivo bmp para o estado passivo
BmpActiveName	Obtém/Define o nome do arquivo bmp para o estado ativo
State	
Pressed	Obtém/Define o estado do controle
Locking	Obtém/Define a propriedade de controle "Locking"
Manipuladores de eventos internos	
OnSetZOrder	Manipulador de evento "SetZOrder"
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"
OnChange	Manipulador de eventos "Change"
OnActivate	Manipulador de evento "Activate"
OnDeactivate	Manipulador de evento "Deactivate"
OnMouseDown	Manipulador de eventos "MouseDown"
OnMouseUp	Manipulador de eventos "MouseUp"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndObj

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com etiqueta bitmap

```
//+------------------------------------------------------------------+
//|                                          ControlsBmpButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                           https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CBmpButton"
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                    // CBmpButton object
   CBmpButton        m_bmpbutton2;                    // CBmpButton object
```

```
public:
                      CControlsDialog(void);
                     ~CControlsDialog(void);
   //--- create
   virtual bool       Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool       OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool               CreateBmpButton1(void);
   bool               CreateBmpButton2(void);
   //--- handlers of the dependent controls events
   void               OnClickBmpButton1(void);
   void               OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- succeed
   return(true);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton1" button                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "BmpButton2" fixed button                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- sets the name of bmp files of the control CBmpButton
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" State of the control is: On");
   else
      Comment(__FUNCTION__+" State of the control is: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

© 2000-2024, MetaQuotes Ltd.

Create

Cria o novo controle CBmpButton.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Border (Método Get)

Obtém a propriedade do controle "Border" (largura da borda).

```
int  Border()  const
```

Valor de retorno

A propriedade "Border" .

Border (Método Set)

Define a propriedade do controle "Border" (largura da borda).

```
bool  Border(
   const int  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade "Border".

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpNames

Define o nome do arquivo bmp de controle

```
bool  BmpNames(
   const string  off="",      // file name
   const string  on=""        // file name
   )
```

Parâmetros

off=""

[in] Nome do arquivo bmp para o estado OFF.

on=""

[in] Nome do arquivo bmp para o estado ON.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpOffName (Método Get)

Obtém o nome do arquivo bmp para o estado OFF.

```
string  BmpOffName()  const
```

Valor de retorno

Nome do arquivo de bmp para o estado OFF.

BmpOffName (Método Set)

Define o nome do arquivo bmp para o estado OFF.

```
bool  BmpOffName(
   const string  name      // file name
   )
```

Parâmetros

name

[in] Nome do arquivo bmp para o estado OFF.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpOnName (Método Get)

Obtém o nome do arquivo bmp para o estado ON.

```
string  BmpOnName()  const
```

Valor de retorno

Nome do arquivo de bmp para o estado ON.

BmpOnName (Método Set)

Define o nome do arquivo bmp para o estado ON.

```
bool  BmpOnName(
   const string  name      // file name
   )
```

Parâmetros

name

[in] Nome do arquivo bmp para o estado ON.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpPassiveName (Método Get)

Obtém o nome do arquivo bmp para o estado passivo de controle.

```
string  BmpPassiveName()  const
```

Valor de retorno

Nome de arquivo bmp para o estado passivo de controle.

BmpPassiveName (Método Set)

Define o nome do arquivo bmp para o estado passivo.

```
bool  BmpPassiveName(
   const string  name      // file name
   )
```

Parâmetros

name

[in] Nome do arquivo bmp para o estado passivo de controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpActiveName (Método Get)

Obtém o nome do arquivo bmp para o estado ativo.

```
string  BmpActiveName()  const
```

Valor de retorno

Nome de arquivo bmp para o estado ativo.

Observação

O controle se torna ativo quando o cursor do mouse está pairando sobre ele.

BmpActiveName (Método Set)

Define o nome do arquivo bmp para o estado ativo.

```
bool  BmpActiveName(
   const string  name      // file name
   )
```

Parâmetros

name

[in] Nome do arquivo bmp para o estado ativo.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Pressed (Método Get)

Obtém o estado do controle (propriedade "Pressed") .

```
bool  Pressed()  const
```

Valor de retorno

Estado de controle.

Pressed (Método Set)

Define o estado do controle (propriedade "Pressed).

```
bool  Pressed(
   const bool  pressed      // new state
   )
```

Parâmetros

pressed

[in] Novo estado de controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Locking (Método Get)

Obtém a propriedade do controle "Locking".

```
bool  Locking()  const
```

Valor de retorno

O valor da propriedade "Locking".

Locking (Método Set)

Define novo valor da propriedade do controle "Locking"

```
void  Locking(
   const bool  locking      // new value
   )
```

Parâmetros

locking

[in] O novo valor da propriedade "Locking".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

O manipulador de eventos do controle "SetZOrder" (alteração da propriedade OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnActivate

O manipulador de eventos do controle "Activate".

```
virtual bool  OnActivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnDeactivate

O manipulador de eventos do controle "Deactivate" .

```
virtual bool  OnDeactivate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

O manipulador de eventos do controle "MouseDown".

```
virtual bool  OnMouseDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseDown" ocorre quando o botão esquerdo do mouse é pressionado no controle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

O manipulador de eventos do controle "MouseUp" (botão esquerdo do mouse liberando o botão).

```
virtual bool  OnMouseUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseUp" ocorre quando o botão esquerdo do mouse é liberado no controle.

© 2000-2024, MetaQuotes Ltd.

CButton

CButton é uma classe do controle simples, baseado no objeto de gráfico "Button".

Descrição

Classe CButton destina-se a criação de botões simples.

Declaração

```
class CButton : public CWndObj
```

Título

```
#include <Controls\Button.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CButton

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
State	
Pressed	Obtém/Define a propriedade "Pressed"
Locking	Obtém/Define a propriedade "Locking"
Propriedades que alteram manipuladores de eventos	
OnSetText	Manipulador de eventos "SetText"
OnSetColor	Manipulador de eventos "SetColor"
OnSetColorBackground	Manipulador de eventos "SetColorBackground"
OnSetColorBorder	Manipulador de eventos "SetColorBorder"
OnSetFont	Manipulador de eventos "SetFont"
OnSetFontSize	Manipulador de eventos "SetFontSize"
Manipuladores de eventos internos	
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"
OnResize	Manipulador de evento "Resize"
OnMouseDown	Manipulador de eventos "MouseDown"
OnOnMouseUp	Manipulador de eventos "MouseUp"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com botão:

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                ControlsButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                       // the button object
   CButton           m_button2;                       // the button object
   CButton           m_button3;                       // the fixed button object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
```

```
protected:
   //--- create dependent controls
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- handlers of the dependent controls events
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Create the "Button1" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button2" button                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Button3" fixed button                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
```

© 2000-2024, MetaQuotes Ltd.

```
//--- create
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" Estado dos controles: On");
   else
      Comment(__FUNCTION__+" Estado dos controles: Off");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
```

© 2000-2024, MetaQuotes Ltd.

```
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CButton.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Pressed (Método Get)

Obtém o estado do controle (propriedade "Pressed") .

```
bool  Pressed()  const
```

Valor de retorno

Estado de controle.

Pressed (Método Set)

Define o estado do controle (propriedade "Pressed).

```
bool  Pressed(
   const bool  pressed      // new state
   )
```

Parâmetros

pressed

[in] Novo estado de controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Locking (Método Get)

Obtém a propriedade do controle "Locking".

```
bool  Locking()  const
```

Valor de retorno

O valor da propriedade "Locking".

Locking (Método Set)

Define novo valor da propriedade de controle "Locking".

```
void  Locking(
   const bool  locking      // new value
   )
```

Parâmetros

locking

[in] O novo valor da propriedade "Locking".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnSetText

O manipulador de eventos do controle "SetText" (altera a propriedade OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

O manipulador de eventos do controle "SetColor" (altera a propriedade OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

O manipulador de eventos do controle "SetColorBackground" (alteração da propriedade OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

O manipulador de eventos do controle "SetColorBorder" (altera propriedadeOBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

O manipulador de eventos do controle "SetFont" (altera propriedade OBJPROP_FONT.

```
virtual bool  OnSetFont()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

O manipulador de eventos do controle "SetFontSize" (altera a propriedade OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMouseDown

O manipulador de eventos do controle "MouseDown".

```
virtual bool  OnMouseDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseDown" ocorre quando o botão esquerdo do mouse é pressionado no controle.

© 2000-2024, MetaQuotes Ltd.

OnMouseUp

O manipulador de eventos do controle "MouseUp" (botão esquerdo do mouse liberando o botão).

```
virtual bool  OnMouseUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "MouseUp" ocorre quando o botão esquerdo do mouse é liberado no controle.

© 2000-2024, MetaQuotes Ltd.

CEdit

CEdit é uma classe do controle simples, baseado no objeto de gráfico "Edit" .

Descrição

Classe CEdit é destinada a criação de controles, onde o usuário pode inserir texto.

Declaração

```
class CEdit : public CWndObj
```

Título

```
#include <Controls\Edit.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CEdit

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriedades	
ReadOnly	Obtém/Define a propriedade "ReadOnly"
TextAlign	Obtém/Define a propriedade "TextAlign"
Manipuladores de eventos de objetos Gráfico	
OnObjectEndEdit	The CHARTEVENT_OBJECT_ENDEDIT event handler (virtual)
Propriedades que alteram manipuladores de eventos	
OnSetText	Manipulador de eventos "SetText"
OnSetColor	Manipulador de eventos "SetColor"
OnSetColorBackground	Manipulador de eventos "SetColorBackground"
OnSetColorBorder	Manipulador de eventos "SetColorBorder"
OnSetFont	Manipulador de eventos "SetFont"
OnSetFontSize	Manipulador de eventos "SetFontSize"
OnSetZOrder	Manipulador de evento "SetZOrder"
Manipuladores de eventos internos	
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"
OnResize	Manipulador de evento "Resize"
OnChange	Manipulador de eventos "Change"
OnClick	Manipulador de eventos "Clique"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWndObj

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com caixa de texto editável:

```
//+------------------------------------------------------------------+
//|                                                ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CEdit             m_edit;                          // CEdit objeto
```

```
public:
                      CControlsDialog(void);
                     ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler

protected:
   //--- create dependent controls
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the display field                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
```

© 2000-2024, MetaQuotes Ltd.

```
//--- permitimos modificar o conteúdo
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam,  // event parameter of the double type
                  const string& sparam)  // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CEdit.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ReadOnly (Método Get)

Obtém a propriedade do controle "ReadOnly".

```
bool  ReadOnly()
```

Valor de retorno

O valor da propriedade "ReadOnly".

ReadOnly (Método Set)

Define o valor da propriedade do controle "ReadOnly".

```
bool  ReadOnly(
   const bool  flag      // new values
   )
```

Parâmetros

flag

[in] O novo valor da propriedade "ReadOnly".

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

TextAlign (Método Get)

Obtém o valor da propriedade do controle "TextAlign" (Alinhamento do modo texto).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Valor de retorno

Valor da propriedade do controle "TextAlign".

TextAlign (Método Set)

Define novo valor da propriedade do controle "TextAlign" (Alinhamento do modo texto).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // new value
   )
```

Parâmetros

align

[in] O novo valor da propriedade "TextAlign".

Valor de retorno

verdadeiro se obteve êxito, falso se a propriedade não foi alterada.

© 2000-2024, MetaQuotes Ltd.

OnObjectEndEdit

O manipulador de evento CHARTEVENT_OBJECT_ENDEDIT.

```
virtual bool  OnObjectEndEdit()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetText

O manipulador de eventos do controle "SetText" (altera a propriedade OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColor

O manipulador de eventos do controle "SetColor" (altera a propriedade OBJPROP_COLOR).

```
virtual bool  OnSetColor()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

O manipulador de eventos do controle "SetColorBackground" (alteração da propriedade OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

O manipulador de eventos do controle "SetColorBorder" (altera propriedade OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFont

O manipulador de eventos do controle "SetFont" (altera propriedade OBJPROP_FONT.

```
virtual bool  OnSetFont()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetFontSize

O manipulador de eventos do controle "SetFontSize" (altera a propriedade OBJPROP_FONTSIZE).

```
virtual bool  OnSetFontSize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetZOrder

O manipulador de eventos do controle "SetZOrder" (alteração da propriedade OBJPROP_ZORDER).

```
virtual bool  OnSetZOrder()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClick

O manipulador de eventos do controle "Click" (clique do botão esquerdo do mouse) .

```
virtual bool  OnClick()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CPanel

CPanel é uma classe de controle simples, baseado no objeto gráfico "Rectangle label".

Descrição

A classe CPanel destina-se a combinar os controles com funções semelhantes no grupo.

Declaração

```
class CPanel : public CWndObj
```

Título

```
#include <Controls\Panel.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CPanel

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriedades do objeto de gráfico	
BorderType	Obtém a propriedade do objeto gráfico "BorderType"
Manipuladores de eventos de objetos Gráfico	
OnSetText	Manipulador de eventos "SetText"
OnSetColorBackground	Manipulador de eventos "SetColorBackground"
OnSetColorBorder	Manipulador de eventos "SetColorBorder"
Manipuladores de eventos internos	
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"
OnResize	Manipulador de evento "Resize"
OnChange	Manipulador de eventos "Change"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com rótulo retangular:

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- create dependent controls
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePanel())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CPanel"                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- coordinates
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- create
   if(!my_white_border.Create(0,ExtDialog.Name()+"MyWhiteBorder",m_subwin,x1,y1,x2,y2)
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // object CPanel
bool     pause=true;             // true - pausa
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//---
   EventSetTimer(3);
   pause=true;
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CPanel.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BorderType (Método Get)

Obtém a propriedade "BorderType" do objeto gráfico.

```
ENUM_BORDER_TYPE  BorderType()
```

Valor de retorno

O valor da propriedade "BorderType".

BorderType (Método Set)

Define novo valor da propriedade "BorderType" do objeto gráfico.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // value
   )
```

Parâmetros

type

[in] O novo valor da propriedade "BorderType".

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnSetText

O manipulador de eventos do controle "SetText" (altera a propriedade OBJPROP_TEXT).

```
virtual bool  OnSetText()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBackground

O manipulador de eventos do controle "SetColorBackground" (alteração da propriedade OBJPROP_BGCOLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnSetColorBorder

O manipulador de eventos do controle "SetColorBorder" (altera propriedade OBJPROP_BORDER_COLOR).

```
virtual bool  OnSetColorBackground()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CPicture

CPicture é uma classe do controle simples, baseado no objeto gráfico "Bitmap Label".

Descrição

A classe CPicture destina-se a criação de imagens gráficas simples.

Declaração

```
class CPicture : public CWndObj
```

Título

```
#include <Controls\Picture.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndObj
      CPicture

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Propriedades do objeto de gráfico	
Border	Obtém/Define a largura da borda do objeto gráfico
BmpName	Obtém/Define o nome de arquivo bmp do controle
Eventos internos	
OnCreate	Manipulador de evento "Create"
OnShow	Manipulador de evento "Show"
OnHide	Manipulador de evento "Hide"
OnMove	Manipulador de evento "Move"
OnChange	Manipulador de eventos "Change"

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Métodos herdados da classe CWnd

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Exemplo de criação de painel com etiqueta bitmap:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
```

```
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                       // CPicture object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreatePicture(void);
   //--- handlers of the dependent controls events
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreatePicture())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "Picture"                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- create
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- definimos o nome dos arquivos bmp para exibir os controles CPicture
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
```

© 2000-2024, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CPicture.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Border (Método Get)

Obtém a propriedade do controle "Border" (largura da borda).

```
int  Border()  const
```

Valor de retorno

A propriedade "Border" .

Border (Método Set)

Define a propriedade do controle "Border" (largura da borda).

```
bool  Border(
   const int  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade "Border".

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BmpName (Método Get)

Obtém o nome de arquivo bmp do controle.

```
string  BmpName()  const
```

Valor de retorno

Nome de arquivo bmp do controle.

BmpName (Método Set)

Define o nome de arquivo bmp do controle.

```
bool  BmpName(
   const string  name      // file name
   )
```

Parâmetros

name

[in] Nome de arquivo bmp do controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnCreate

O manipulador de eventos do controle "Create".

```
virtual bool  OnCreate()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnMove

O manipulador de eventos do controle "Move".

```
virtual bool  OnMove()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChange

O manipulador de eventos do controle "Change".

```
virtual bool  OnChange()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CScroll

CScroll é uma classe base para a criação de barras de rolagem.

Descrição

A classe CScroll é um controle complexo (com controles dependentes), contém a funcionalidade básica para criação de barras de rolagem. A própria classe de base não é utilizada como um controle separado, dois dos seus herdeiros (as classes CScrollV e CScrollH) são utilizados como controles.

Declaração

```
class CScroll : public CWndContainer
```

Título

```
#include <Controls\Scrolls.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CScroll

Descendentes diretos

CScrollH, CScrollV

Métodos de classe

Create	
Create	Criar controle
Manipuladores de eventos de objetos Gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
MinPos	Obtém/Define a posição mínima
MaxPos	Obtém/Define a posição máxima
CurrPos	Obtém/Define a posição atual
Criação de controles dependentes	
CreateBack	Cria botão de segundo plano (fundo)
CreateInc	Cria botão de aumentar a barra de rolagem

© 2000-2024, MetaQuotes Ltd.

Create	
CreateDec	Cria botão de diminuir a barra de rolagem
CreateThumb	Cria botão polegar (pode ser arrastado) da barra de rolagem
Manipuladores de controle de eventos dependentes	
OnClickInc	Manipulador de eventos, utilizados para a manipulação de eventos de botão de aumentar
OnClickDec	Manipulador de eventos, utilizados para a manipulação de eventos de botão de diminuir
Manipuladores de eventos internos	
OnShow	Manipulador de evento "Create"
OnHide	Manipulador de evento "Hide"
OnChangePos	Manipulador de evento "ChangePosition"
Manipuladores para arrastar objetos	
OnThumbDragStart	Manipulador de evento "ThumbDragStart"
OnThumbDragProcess	Manipulador de evento "ThumbDragProcess"
OnThumbDragEnd	Manipulador de evento "ThumbDragEnd"
Posição	
CalcPos	Obtém posição da barra de rolagem pela coordenada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CScroll.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

MinPos (Método Get)

Obtém o valor da "MinPos" (posição mínima) do controle CScroll.

```
int  MinPos()  const
```

Valor de retorno

Novo valor da propriedade "MinPos".

MinPos (Método Set)

Define o valor de "MinPos" (posição mínima) do controle CScroll.

```
void  MinPos(
   const int  value      // new value
   )
```

Parâmetros

value

[in] Novo valor da propriedade "MinPos".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MaxPos (Get method)

Obtém o valor da "MaxPos" (posição máxima) do controle CScroll.

```
int  MaxPos()  const
```

Valor de retorno

Novo valor da propriedade "MaxPos".

MaxPos (Set method)

Define o valor da "MaxPos" (posição máxima) do controle CScroll.

```
void  MaxPos(
   const int  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade "MaxPos".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CurrPos (Get method)

Obtém o valor da "CurrPos" (posição atual) do controle CScroll.

```
int  CurrPos()  const
```

Valor de retorno

Novo valor da propriedade "CurrPos".

CurrPos (Set method)

Define o valor de "CurrPos" (posição atual) do controle CScroll.

```
void  CurrPos(
   const int  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade "CurrPos".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CreateBack

Cria botão de segundo plano (fundo) do controle CScroll.

```
virtual bool  CreateBack()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateInc

Cria botão de aumentar do controle CScroll.

```
virtual bool  CreateInc()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateDec

Cria botão de diminuir do controle CScroll.

```
virtual bool  CreateDec()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Cria botão polegar (pode ser arrastado) do controle CScroll.

```
virtual bool  CreateThumb()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickInc

O manipulador de eventos do controle "ClickInc" (clique no botão esquerdo do mouse sobre o botão de aumentar).

```
virtual bool  OnClickInc()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClickDec

O manipulador de eventos do controle "ClickDec" (clique no botão esquerdo do mouse sobre o botão de diminuir).

```
virtual bool  OnClickDec()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnShow

O manipulador de eventos do controle "Show".

```
virtual bool  OnShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnHide

O manipulador de eventos do controle "Hide".

```
virtual bool  OnHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

O manipulador de eventos do controle "ChangePos" (mudança de posição).

```
virtual bool  OnChangePos()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

O manipulador de eventos do controle "ThumbDragStart" (início para arrastar).

```
virtual bool  OnThumbDragStart()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "ThumbDragStart" ocorre no início da operação de arrastar.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

O manipulador de eventos do controle "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parâmetros

x

[in] Coordenada X atual do cursor de mouse.

y

[in] Coordenada Y atual do cursor do mouse.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragProcess" ocorre quando o controle da barra de rolagem (botão polegar) é movido.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

O manipulador de eventos do controle "ThumbDragEnd" (finalizado o processo de arrastar).

```
virtual bool  OnThumbDragEnd()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragEnd" ocorre quando a operação de arrastar do controle da barra de rolagem (botão polegar) estiver concluída.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Obtém posição da barra de rolagem pela coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parâmetros

coord

[in] Barra de rolagem de coordenadas.

Valor de retorno

Posição da barra de rolagem.

© 2000-2024, MetaQuotes Ltd.

CScrollV

CScrollV é uma classe de controle complexa da "barra de rolagem vertical".

Descrição

Classe CScrollV destina-se a criação de barras de rolagem vertical.

Declaração

```
class CScrollV : public CScroll
```

Título

```
#include <Controls\Scrolls.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CScroll
        CScrollV

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Controles dependentes	
CreateInc	Cria botão de aumentar a barra de rolagem
CreateDec	Cria botão de diminuir a barra de rolagem
CreateThumb	Cria o botão polegar da barra de rolagem (pode ser arrastado)
Manipuladores de eventos internos	
OnResize	Manipulador de evento "Resize"
OnChangePos	Manipulador de evento "ChangePosition"
Manipulador de eventos para arrastar	
OnThumbDragStart	Manipulador de evento "ThumbDragStart"
OnThumbDragProcess	Manipulador de evento "ThumbDragProcess"
OnThumbDragEnd	Manipulador de evento "ThumbDragEnd"
Posição	
CalcPos	Obtém posição da barra de rolagem pela coordenada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Métodos herdados da classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Exemplo de criação de painel com rolagem vertical:

```
//+------------------------------------------------------------------+
//|                                              ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                     // CScrollV objeto

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateScrollV(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollV())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsV object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam,  // event parameter of the double type
                  const string& sparam)  // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateInc

Cria botão de aumentar do controle.

```
virtual bool  CreateInc()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateDec

Cria botão de diminuir do controle.

```
virtual bool  CreateDec()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Cria botão polegar (pode ser arrastado) do controle.

```
virtual bool  CreateThumb()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

O manipulador de eventos do controle "ChangePos" (mudança de posição).

```
virtual bool  OnChangePos()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

O manipulador de eventos do controle "ThumbDragStart" (início para arrastar).

```
virtual bool  OnThumbDragStart()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "ThumbDragStart" ocorre no início da operação de arrastar.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

O manipulador de eventos do controle "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parâmetros

x

[in] Coordenada X atual do cursor de mouse.

y

[in] Coordenada Y atual do cursor do mouse.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragProcess" ocorre quando o controle da barra de rolagem (botão polegar) é movido.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

O manipulador de eventos do controle "ThumbDragEnd" (finalizado o processo de arrastar).

```
virtual bool  OnThumbDragEnd()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragEnd" ocorre quando a operação de arrastar do controle da barra de rolagem (botão polegar) estiver concluída.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Obtém posição da barra de rolagem pela coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parâmetros

coord

[in] Barra de rolagem de coordenadas.

Valor de retorno

Posição da barra de rolagem.

© 2000-2024, MetaQuotes Ltd.

CScrollH

CScrollH é uma classe de controle complexa da "barra de rolagem horizontal" .

Descrição

CScrollH destina-se a criação de barras de rolagem horizontal.

Declaração

```
class CScrollH : public CScroll
```

Título

```
#include <Controls\Scrolls.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CScroll
        CScrollH

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Controles dependentes	
CreateInc	Cria botão de aumentar a barra de rolagem
CreateDec	Cria botão de diminuir a barra de rolagem
CreateThumb	Cria o botão polegar da barra de rolagem (pode ser arrastado)
Manipuladores de eventos internos	
OnResize	Manipulador de evento "Resize"
OnChangePos	Manipulador de evento "ChangePosition"
Manipulador de eventos para arrastar	
OnThumbDragStart	Manipulador de evento "ThumbDragStart"
OnThumbDragProcess	Manipulador de evento "ThumbDragProcess"
OnThumbDragEnd	Manipulador de evento "ThumbDragEnd"
Posição	
CalcPos	Obtém posição da barra de rolagem pela coordenada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Métodos herdados da classe CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Exemplo de criação de painel com rolagem horizontal:

```
//+------------------------------------------------------------------+
//|                                              ControlsScrollH.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                     // CScrollH objeto

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateScrollsH(void);
   //--- handlers of the dependent controls events
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateScrollsH())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the CScrollsH object                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- create
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- set up the scrollbar
   m_scroll_v.MinPos(0);
//--- set up the scrollbar
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Posição da barra de rolagem",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
void OnChartEvent(const int id,          // event ID
                  const long& lparam,    // event parameter of the long type
                  const double& dparam,  // event parameter of the double type
                  const string& sparam)  // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

CreateInc

Cria botão de aumentar do controle.

```
virtual bool  CreateInc()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateDec

Cria botão de diminuir do controle.

```
virtual bool  CreateDec()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateThumb

Cria botão polegar (pode ser arrastado) do controle.

```
virtual bool  CreateThumb()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangePos

O manipulador de eventos do controle "ChangePos" (mudança de posição).

```
virtual bool  OnChangePos()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragStart

O manipulador de eventos do controle "ThumbDragStart" (início para arrastar).

```
virtual bool  OnThumbDragStart()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "ThumbDragStart" ocorre no início da operação de arrastar.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragProcess

O manipulador de eventos do controle "ThumbDragProcess".

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x coordinate
   const int  y       // y coordinate
   )
```

Parâmetros

x

[in] Coordenada X atual do cursor de mouse.

y

[in] Coordenada Y atual do cursor do mouse.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragProcess" ocorre quando o controle da barra de rolagem (botão polegar) é movido.

© 2000-2024, MetaQuotes Ltd.

OnThumbDragEnd

O manipulador de eventos do controle "ThumbDragEnd" (finalizado o processo de arrastar).

```
virtual bool  OnThumbDragEnd()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O "ThumbDragEnd" ocorre quando a operação de arrastar do controle da barra de rolagem (botão polegar) estiver concluída.

© 2000-2024, MetaQuotes Ltd.

CalcPos

Obtém posição da barra de rolagem pela coordenada.

```
virtual int  CalcPos(
   const int      coord      // coordinate
   )
```

Parâmetros

coord

[in] Barra de rolagem de coordenadas.

Valor de retorno

Posição da barra de rolagem.

© 2000-2024, MetaQuotes Ltd.

CWndClient

CWndClient é uma classe de controle complexo da "área de cliente" (com controles dependentes). É uma classe base para criação de área de barras de rolagem.

Descrição

CWndClient implementa a funcionalidade para criação de área de barras de rolagem.

Declaração

```
class CWndClient : public CWndContainer
```

Título

```
#include <Controls\WndClient.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CWndClient

Descendentes diretos

CCheckGroup, CListView, CRadioGroup

Métodos de classe

Create	
Create	Criar controle
Manipulador de eventos de gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
ColorBackground	Define a cor do segundo plano (fundo)
ColorBorder	Define a cor da borda
BorderType	Define o tipo da borda
Configuração	
VScrolled	Obtém/define um flag, indicando que a barra de rolagem vertical está sendo usada
HScrolled	Obtém/Define o flag, indicando que a barra de rolagem horizontal está sendo usada

© 2000-2024, MetaQuotes Ltd.

Create	
Controles dependentes	
CreateBack	Cria segundo plano (fundo) para a barra de rolagem
CreateScrollV	Cria a barra de rolagem vertical
CreateScrollH	Cria a barra de rolagem horizontal
Manipuladores de eventos internos	
OnResize	Manipulador de evento "Resize"
Manipuladores de controle de eventos dependentes	
OnVScrollShow	Manipulador de eventos "Show" (virtual), dependente do controle VScroll
OnVScrollHide	Manipulador de eventos "Hide" (virtual), dependente do controle VScroll
OnHScrollShow	Manipulador de eventos "Show" (virtual), dependente do controle HScroll
OnHScrollHide	Manipulador de eventos "Hide" (virtual), dependente do controle HScroll
OnScrollLineDown	Manipulador de eventos "ScrollLineDown" (virtual), dependente do controle VScroll
OnScrollLineUp	Manipulador de eventos "ScrollLineUp" (virtual), dependente do controle VScroll
OnScrollLineLeft	Manipulador de eventos "ScrollLineLeft" (virtual), dependente do controle de HScroll
OnScrollLineRight	Manipulador de eventos "ScrollLineRight" (virtual), dependente do controle de HScroll
Resize	
Rebound	Define novas coordenadas do controle usando coordenadas de classe CRect

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CWndClient.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

ColorBackground

Define a cor do segundo plano (fundo) do controle.

```
bool  ColorBackground(
   const color  value      // new color
   )
```

Parâmetros

value

[in] Nova cor do segundo plano (fundo) do controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ColorBorder

Definir a cor da borda do controle.

```
bool  ColorBorder(
   const color  value      // color
   )
```

Parâmetros

value

[in] Nova cor da borda do controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

BorderType

Define o tipo da borda do controle.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // border type
   )
```

Parâmetros

type

[in] Tipo da borda do controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

VScrolled (Método Get)

Obtém um flag, indicando que barra de rolagem vertical está sendo usada.

```
bool  VScrolled()
```

Valor de retorno

verdadeiro, se barra de rolagem vertical está sendo usada, caso contrário é falso.

VScrolled (Método Set)

Define um flag, indicando que barra de rolagem vertical está sendo usada

```
bool  VScrolled(
   const bool  flag      // flag
   )
```

Parâmetros

flag

[in] Flag.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

HScrolled (Método Get)

Obtém um flag, indicando que barra de rolagem horizontal está sendo usada.

```
bool  HScrolled()
```

Valor de retorno

verdadeiro, se barra de rolagem horizontal está sendo usada, caso contrário é falso.

HScrolled (Método Set)

Define um flag, indicando que barra de rolagem horizontal está sendo usada

```
bool  HScrolled(
   const bool  flag      // flag
   )
```

Parâmetros

flag

[in] Flag.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateBack

Cria botão de segundo plano (fundo) do controle.

```
virtual bool  CreateBack()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateScrollV

Cria barra de rolagem vertical.

```
virtual bool  CreateScrollV()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateScrollH

Cria barra de rolagem horizontal.

```
virtual bool  CreateScrollH()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

O manipulador de eventos do controle "VScrollShow" (barra de rolagem vertical exibida).

```
virtual bool  OnVScrollShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

O manipulador de eventos do controle "VScrollHide" (barra de rolagem vertical oculta).

```
virtual bool  OnVScrollHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnHScrollShow

O manipulador de eventos do controle "HScrollShow" (barra de rolagem horizontal show).

```
virtual bool  OnHScrollShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnHScrollHide

O manipulador de eventos do controle "HScrollHide" (barra de rolagem vertical esconder).

```
virtual bool  OnHScrollHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

O manipulador de eventos do controle "ScrollLineDown" (linha de rolagem vertical para baixo).

```
virtual bool  OnScrollLineDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

O manipulador de eventos do controle "ScrollLineUp" (rolagem vertical linha para cima) .

```
virtual bool  OnScrollLineUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineLeft

O manipulador de eventos do controle "ScrollLineLeft" (rolagem horizontal da linha a esquerda).

```
virtual bool  OnScrollLineLeft()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineRight

O manipulador de evento do controle "ScrollLineRight" (rolagem horizontal da linha a direita).

```
virtual bool  OnScrollLineRight()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O método da classe base não faz nada e sempre retorna verdadeiro.

© 2000-2024, MetaQuotes Ltd.

ReBound

Define novas coordenadas do controle usando as coordenadas da classe CRect.

```
void  ReBound(
   const & CRect  rect      // CRect class
   )
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CListView

CListView é uma classe de controle complexa (com controles dependentes).

Descrição

A classe CListView encapsula a funcionalidade do controle de lista.

Declaração

```
class CListView : public CWndClient
```

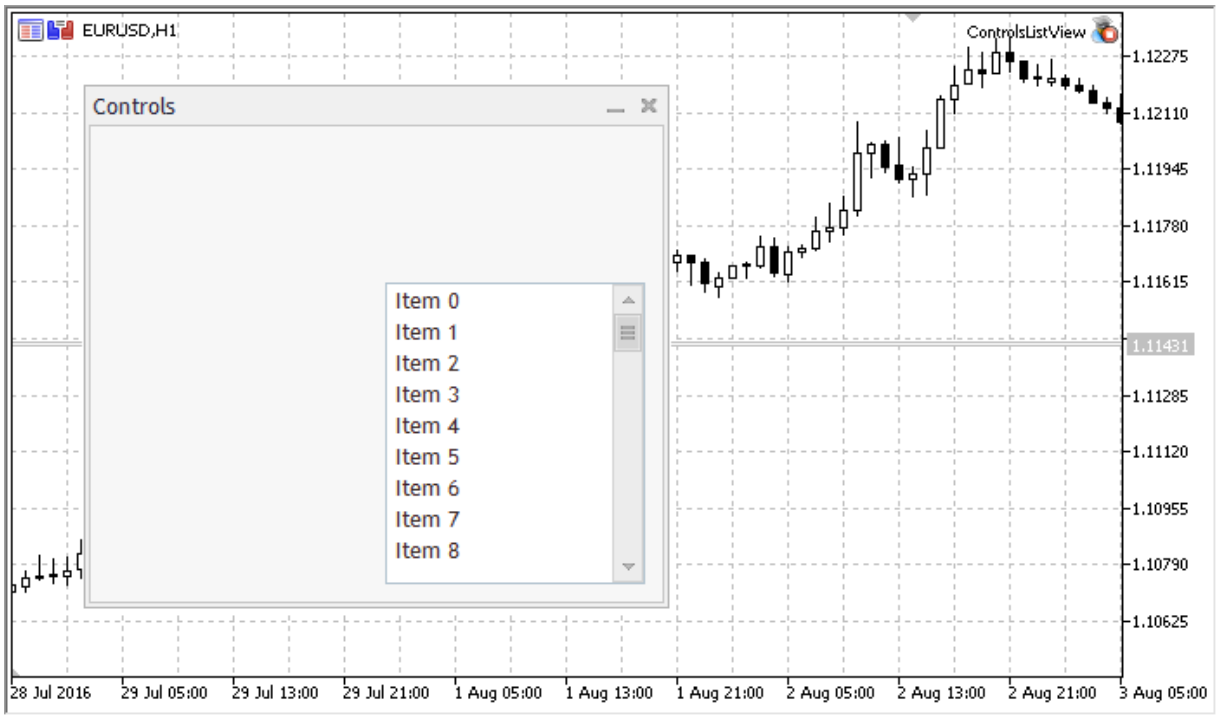
Título

```
#include <Controls\ListView.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CWndClient
        CListView

O resultado de trabalho é apresentado abaixo do código:



Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Configuração	
TotalView	Define o número de itens, exibido no controle
Adicionar/Deletar	
AddItem	Adiciona um item
Dados	
Select	Seleciona atual item da lista pelo índice
SelectByText	Seleciona item da lista atual pelo texto
SelectByValue	Seleciona atual item da lista pelo valor
Somente leitura de dados	
Value	Obtém o valor do item atual da lista
Controles dependentes	
CreateRow	Cria uma linha da ListView
Manipuladores de eventos internos	
OnResize	Manipulador de eventos "Resize" (virtual)
Manipuladores de controle de eventos dependentes	
OnVScrollShow	Manipulador de eventos "Show" (virtual), dependente do controle VScroll
OnVScrollHide	Manipulador de eventos "Hide" (virtual), dependente do controle VScroll
OnScrollLineDown	Manipulador de eventos "ScrollLineDown" (virtual), dependente do controle VScroll
OnScrollLineUp	Manipulador de eventos "ScrollLineUp" (virtual), dependente do controle VScroll
OnItemClick	Manipulador de eventos "ItemClick" (virtual)
Redraw	
Redraw	edesenha o controle
RowState	Define o estado da linha determinada

© 2000-2024, MetaQuotes Ltd.

Create	
CheckView	Verifica a visibilidade da linha determinada

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Métodos herdados da classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Exemplo de criação de painel com lista de valores:

```
//+------------------------------------------------------------------+
//|                                              ControlsListView.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
```

```
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // CListView objeto

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateListView(void);
   //--- handlers of the dependent controls events
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateListView())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "ListView" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- create
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CListView.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

TotalView

Define o número de itens, exibido no controle.

```
bool  TotalView(
   const int  value      // items shown
   )
```

Parâmetros

value

[in] O número de itens, exibido no controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Observação

O número de itens exibidos pode ser determinado apenas uma vez.

© 2000-2024, MetaQuotes Ltd.

AddItem

Adiciona um item ao controle.

```
bool  AddItem(
   const string  item,      // text
   const long    value      // value
   )
```

Parâmetros

item

[in] Texto.

value

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Select

Seleciona atual item da lista pelo índice.

```
bool  Select(
   const int  index     // index
   )
```

Parâmetros

index

[in] Índice do item.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SelectByText

Seleciona atual item da lista pelo texto.

```
bool  SelectByText(
   const string  text     // text
   )
```

Parâmetros

text

[in] Texto.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SelectByValue

Seleciona atual item da lista pelo valor.

```
bool  SelectByValue(
   const long  value     // value
   )
```

Parâmetros

value

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Value

Obtém o valor do item atual da lista.

```
long  Value()
```

Valor de retorno

O valor do item atual da lista.

© 2000-2024, MetaQuotes Ltd.

CreateRow

Cria uma linha de controle "CListView".

```
bool  CreateRow(
   const int  index     // index
   )
```

Parâmetros

index

[in] Índice do item.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnResize

O manipulador de eventos do controle "Resize" .

```
virtual bool  OnResize()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

O manipulador de eventos do controle "VScrollShow" (barra de rolagem vertical exibida).

```
virtual bool  OnVScrollShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

O manipulador de eventos do controle "VScrollHide" (barra de rolagem vertical oculta).

```
virtual bool  OnVScrollHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

O manipulador de eventos do controle "ScrollLineDown" (linha de rolagem vertical para baixo).

```
virtual bool  OnScrollLineDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

O manipulador de eventos do controle "ScrollLineUp" (rolagem vertical linha para cima) .

```
virtual bool  OnScrollLineUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnItemClick

O manipulador de eventos do controle "ItemClick" (botão do mouse clica numa linha).

```
virtual bool  OnItemClick()
   const int    index     // row index
   )
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redesenha o controle.

```
bool  Redraw()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

RowState

Define o estado da linha determinada.

```
bool  RowState(
   const int   index      // index
   const bool  select     // state
   )
```

Parâmetros

index

[in] Índice da linha.

select

[in] Estado da linha.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CheckView

Verifica a visibilidade da linha determinada.

```
bool  CheckView()
```

Valor de retorno

verdadeiro, se a linha selecionada é visível, caso contrário é falso.

© 2000-2024, MetaQuotes Ltd.

CComboBox

CComboBox é uma classe do controle complexa (com controles dependentes).

Descrição

ComboBox é constituída por uma caixa de listagem, combinado com um controle estático destinada a seleção. A porção da caixa de listagem do controle pode ser suspensa quando o usuário seleciona o comando ao lado do controle.

Declaração

```
class CComboBox : public CWndContainer
```

Título

```
#include <Controls\ComboBox.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CComboBox

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Add	
AddItem	Adiciona um item
Configuração	
ListViewItems	Define o número de itens, exibido no controle
Dados	
Select	Seleciona atual item da lista pelo índice
SelectByText	Seleciona item da lista atual pelo texto
SelectByValue	Seleciona atual item da lista pelo valor
Somente leitura de dados	
Value	Obtém o valor do atual item da lista
Controles dependentes	
CreateEdit	Cria um controle dependente (editar)
CreateButton	Cria controle dependente (botão)
CreateList	Cria controle dependente (visualizar lista)
Manipuladores de controle de eventos dependentes	
OnClickEdit	Manipulador de eventos "ClickEdit" (virtual)
OnClickButton	Manipulador de eventos "ClickButton" (virtual)
OnChangeList	Manipulador de eventos "Changelist" (virtual)
Exibir/Ocultar	
ListShow	Exibe a lista de itens
ListHide	Oculta a lista de itens

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet,

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Exemplo de criação do painel com controles "Campo com lista suspensa":

```
//+------------------------------------------------------------------+
//|                                              ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CComboBox"
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
```

```
private:
   CComboBox         m_combo_box;;                    // CComboBox object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateComboBox(void);
   //--- handlers of the dependent controls events
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateComboBox())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Create the "ComboBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
           (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- fill out with strings
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CComboBox.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

AddItem

Adiciona um item ao controle.

```
bool  AddItem(
   const string  item,      // text
   const long    value      // value
   )
```

Parâmetros

item

[in] Texto.

value=0

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ListViewItems

Define o número de itens da lista do controle CComboBox.

```
void  ListViewItems(
   const int    value     // number of list items
   )
```

Parâmetros

value

[in] Número de itens da lista.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Select

Seleciona atual item da lista pelo índice.

```
bool  Select(
   const int  index     // index
   )
```

Parâmetros

index

[in] Índice do item.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SelectByText

Seleciona atual item da lista pelo texto.

```
bool  SelectByText(
   const string  text     // text
   )
```

Parâmetros

text

[in] Texto do item.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

SelectByValue

Seleciona atual item da lista pelo valor.

```
bool  SelectByValue(
   const long  value     // value
   )
```

Parâmetros

value

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Value

Obtém o valor do item da lista atual.

```
long  Value()
```

Valor de retorno

O valor do item da lista atual.

© 2000-2024, MetaQuotes Ltd.

CreateEdit

Cria controle dependente (editar) do controle.

```
virtual bool  CreateEdit()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Cria controle dependente (botão).

```
virtual bool  CreateButton()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateList

Cria controle dependente (ver lista).

```
virtual bool  CreateList()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickEdit

O manipulador de enventos do controle "ClickEdit" (clique do mouse sobre o editar).

```
virtual bool  OnClickEdit()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClickButton

O manipulador de eventos do controle "ClickButton" (clique do mouse sobre o botão)

```
virtual bool  OnClickButton()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangeList

O manipulador de eventos "ChangeList" (alteração da lista) .

```
virtual bool  OnChangeList()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

ListShow

Exibe a lista de itens.

```
virtual bool  ListShow()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ListHide

Oculta a lista de itens.

```
virtual bool  ListHide()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CCheckBox

CCheckBox é uma classe de controle complexa.

Descrição

A classe de controle CCheckBox exibe uma caixa de seleção que permite ao usuário selecionar uma condição verdadeira ou falsa.

Declaração

```
class CCheckBox : public CWndContainer
```

Título

```
#include <Controls\CheckBox.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CCheckBox

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

Create	
Create	Criar controle

© 2000-2024, MetaQuotes Ltd.

Create	
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
Text	Obtém/Define a etiqueta de texto associado com o controle
Color	Obtém/Define a cor da etiqueta de texto associado com o controle
State	
Checked	Obtém/Define um valor indicando se o controle está verificado
Dados	
Value	Obtém/Define o valor associado com o controle
Controles dependentes	
CreateButton	Cria controle dependente (botão)
CreateLabel	Cria controle dependente (etiqueta)
Manipuladores de controle de eventos dependentes	
ClickButton	Manipulador de eventos "ClickButton" (virtual)
ClickLabel	Manipulador de eventos "ClickLabel" (virtual)

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Exemplo de criação do painel com controles "Alternador com confirmação":

```
//+------------------------------------------------------------------+
//|                                              ControlsCheckBox.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckBox"
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // CCheckBox object
   CCheckBox         m_check_box2;                    // CCheckBox object
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- create dependent controls
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckBox1(void);
```

```
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
```

© 2000-2024, MetaQuotes Ltd.

```
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckBox" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- coordinates
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- create
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box1.Checked()));
  }
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CCheckBox.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Text (Método Get)

Obtém o texto da etiqueta associado com o controle.

```
string  Text()
```

Valor de retorno

Texto da etiqueta.

Text (Método Set)

Define o texto da etiqueta associado com o controle.

```
bool  Text(
   const string  value      // text
   )
```

Parâmetros

value

[in] O novo texto da etiqueta.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Color (Método Get)

Obtém a cor da etiqueta associada com o controle.

```
color  Color()  const
```

Valor de retorno

Cor da etiqueta.

Color (Método Set)

Define a cor da etiqueta associada com o controle.

```
bool  Color(
   const color  value      // color
   )
```

Parâmetros

value

[in] Nova cor da etiqueta.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Checked (Método Get)

Obtém o estado do controle.

```
bool  Checked()  const
```

Valor de retorno

Estado do controle.

Checked (Método Set)

Define o estado do controle.

```
bool  Checked(
   const bool  flag      // state
   )
```

Parâmetros

flag

[in] Novo estado.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Value (Método Get)

Obtém o valor associado com o controle.

```
int  Value()  const
```

Valor de retorno

O valor associado com o controle.

Value (Método Set)

Define o valor associado com o controle.

```
void  Value(
   const int  value      // new value
   )
```

Parâmetros

value

[in] Novo Valor.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Cria controle dependente (botão).

```
virtual bool  CreateButton()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateLabel

Cria controle dependente (etiqueta).

```
virtual bool  CreateLabel()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickButton

O manipulador de eventos do controle "ClickButton" (clique do mouse sobre o botão)

```
virtual bool  OnClickButton()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClickLabel

O manipulador de eventos do controle "ClickLabel" (clique do mouse sobre a etiqueta).

```
virtual bool  OnClickLabel()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CCheckGroup

CCheckGroup é uma classe de controle complexa (com controles dependentes).

Descrição

CCheckGroup prevê a possibilidade da criação de controles que permitem exibir e editar flags.

Declaração

```
class CCheckGroup : public CWndClient
```

Título

```
#include <Controls\CheckGroup.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos para todos os eventos de gráfico
Add	
AddItem	Adiciona novo item
Somente leitura de dados	
Value	Obtém o valor, associado com o controle
Controles dependentes	
CreateButton	Cria novo item CCheckBox
Manipuladores de controle de eventos dependentes	
OnVScrollShow	Manipulador de eventos "Show" (virtual), dependente do controle VScroll
OnVScrollHide	Manipulador de eventos "Hide" (virtual), dependente do controle VScroll
OnScrollLineDown	Manipulador de eventos "ScrollLineUp" (virtual), dependente do controle VScroll
OnScrollLineUp	Manipulador de eventos "ScrollLineDown" (virtual), dependente do controle VScroll
OnChangeItem	Manipulador de eventos "ChangeItem" (virtual) dependente do controle VScroll
Redraw	
Redraw	Redesenha o grupo
RowState	Define o estado do item especificado

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Métodos herdados da classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Exemplo de criação do painel com controles "Alternador com confirmação independe":

```
//+------------------------------------------------------------------+
//|                                          ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Control Panels and Dialogs. Demonstration class CCheckGroup"
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

```
   CCheckGroup       m_check_group;                    // CCheckGroup object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateCheckGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateCheckGroup())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "CheckGroup" element                                  |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
           (EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+CHECK_HEIGHT;
//--- create
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
```

© 2000-2024, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CCheckGroup.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

AddItem

Adiciona um item ao controle.

```
bool  AddItem(
   const string  item,      // text
   const long    value      // value
   )
```

Parâmetros

item

[in] Texto.

value=0

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Value

Obtém o valor associado com o controle.

```
long  Value()
```

Valor de retorno

O valor, associado com o controle.

Observação

O valor depende do estado de todos os itens do CCheckGroup.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Cria uma nova instância de classe CCheckBox no índice especificado.

```
bool  CreateButton(
   int  index      // index
   )
```

Parâmetros

index

[in] Índice do novo item no CCheckGroup.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

O manipulador de eventos do controle "VScrollShow" (barra de rolagem vertical exibida).

```
virtual bool  OnVScrollShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

O manipulador de eventos do controle "VScrollHide" (barra de rolagem vertical oculta).

```
virtual bool  OnVScrollHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

O manipulador de eventos do controle "ScrollLineDown" (linha de rolagem vertical para baixo).

```
virtual bool  OnScrollLineDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

O manipulador de eventos do controle "ScrollLineUp" (rolagem vertical linha para cima) .

```
virtual bool  OnScrollLineUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangeItem

O manipulador de eventos do controle "ChangeItem" (altera item).

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Parâmetros

index

[in] O índice do item alterado.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redesenha o controle.

```
bool  Redraw()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

RowState

Define o estado do item especificado.

```
bool  RowState(
   const int   index,       // item index
   const bool  select       // state
   )
```

Parâmetros

index

[in] Índice do item para alterar.

select

[in] Novo estado.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CRadioButton

CRadioButton é uma classe controle complexa.

Descrição

A classe CRadioButton em si não é usada, é utilizada para a criação de itens CRadioGroup.

Declaração

```
class CRadioButton : public CWndContainer
```

Título

```
#include <Controls\RadioButton.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CRadioButton

Métodos de classe

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos para todos os eventos de gráfico
Propriedades	
Text	Obtém/Define a etiqueta de texto, associada com o controle
Color	Obtém/Define a cor da etiqueta de texto, associada com o controle
State	
State	Obtém/Define o estado
Controles dependentes	
CreateButton	Cria botão
CreateLabel	Cria etiquetas
Manipuladores de controle de eventos dependentes	
OnClickButton	Manipulador de eventos "ClickButton" (virtual)

© 2000-2024, MetaQuotes Ltd.

Create	
OnClickLabel	Manipulador de eventos "ClickLabel" (virtual)

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CRadioButton.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Text (Método Get)

Obtém o texto da etiqueta associado com o controle.

```
string  Text()
```

Valor de retorno

Texto da etiqueta.

Text (Método Set)

Define o texto da etiqueta associado com o controle.

```
bool  Text(
   const string  value      // text
   )
```

Parâmetros

value

[in] O novo texto da etiqueta.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Color (Método Get)

Obtém a cor da etiqueta associada com o controle.

```
color Color() const
```

Valor de retorno

Cor da etiqueta.

Color (Método Set)

Define a cor da etiqueta associada com o controle.

```
bool Color(
   const color value      // color
   )
```

Parâmetros

value

[in] Nova cor da etiqueta.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

State (Método Get)

Obtém o estado do botão.

```
bool State() const
```

Valor de retorno

Estado do botão.

State (Método Set)

Define o estado do botão.

```
bool State(
   const bool flag      // flag
   )
```

Parâmetros

flag

[in] Novo estado do botão.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Cria botão.

```
virtual bool  CreateButton()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateLabel

Cria etiqueta.

```
virtual bool  CreateLabel()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickButton

O manipulador de eventos do controle "ClickButton" (clique de mouse).

```
virtual bool  OnClickButton()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClickLabel

O manipulador de eventos do controle "ClickLabel" (clique do mouse sobrea etiqueta).

```
virtual bool  OnClickLabel()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CRadioGroup

CRadioGroup é uma classe do controle complexa (com controles dependentes).

Descrição

CRadioGroup permite ao usuário selecionar uma única opção a partir de um grupo de opções quando emparelhado com outros controles CRadioButton.

Declaração

```
class CRadioGroup : public CWndClient
```

Título

```
#include <Controls\RadioGroup.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CWndClient
        CRadioGroup

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Add	
AddItem	Adiciona novo item
Somente leitura de dados	
Value	Obtém o valor, associado com o controle
Controles dependentes	
CreateButton	Cria novo item CRadioButton
Manipuladores de controle de eventos dependentes	
OnVScrollShow	Manipulador de eventos "Show" (virtual), dependente do controle VScroll
OnVScrollHide	Manipulador de eventos "Hide" (virtual), dependente do controle VScroll
OnScrollLineDown	Manipulador de eventos "ScrollLineDown" (virtual), dependente do controle VScroll
OnScrollLineUp	Manipulador de eventos "ScrollLineUp" (virtual), dependente do controle VScroll
OnChangeItem	Manipulador de eventos "ChangeItem" (virtual)
Redraw	
Redraw	Redesenha os itens do grupo
RowState	Define o estado do item especificado
Select	Seleciona o item atual

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Métodos herdados da classe CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Exemplo de criação de painel com grupo "radio-controle":

```
//+------------------------------------------------------------------+
//|                                          ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

```
   CRadioGroup       m_radio_group;                    // CRadioGroup objeto

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
   bool              CreateRadioGroup(void);
   //--- handlers of the dependent controls events
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateRadioGroup())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "RadioGroup" element                                  |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
           (EDIT_HEIGHT+CONTROLS_GAP_Y)+
           (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- create
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- fill out with strings
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
```

© 2000-2024, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CRadioGroup.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

AddItem

Adiciona um item ao controle.

```
bool  AddItem(
   const string  item,      // text
   const long    value=0    // value
   )
```

Parâmetros

item

[in] Texto.

value=0

[in] Valor do tipo long.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Value

Obtém o valor associado com o controle.

```
long  Value()
```

Valor de retorno

O valor, associado com o controle.

Observação

O valor depende do estado de todos os itens do controle CRadioGroup.

© 2000-2024, MetaQuotes Ltd.

CreateButton

Cria uma nova instância de classe CRadioButton no índice determinado.

```
bool  CreateButton(
   const int  index      // index
   )
```

Parâmetros

index

[in] Índice do novo item na CRadioGroup.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollShow

O manipulador de eventos do controle "VScrollShow" (barra de rolagem vertical exibida).

```
virtual bool  OnVScrollShow()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnVScrollHide

O manipulador de eventos do controle "VScrollHide" (barra de rolagem vertical oculta).

```
virtual bool  OnVScrollHide()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineDown

O manipulador de eventos do controle "ScrollLineDown" (linha de rolagem vertical para baixo).

```
virtual bool  OnScrollLineDown()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnScrollLineUp

O manipulador de eventos do controle "ScrollLineUp" (rolagem vertical linha para cima) .

```
virtual bool  OnScrollLineUp()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangeItem

O manipulador de eventos do controle "ChangeItem" (altera item).

```
virtual bool  OnChangeItem(
   const int  index      // index
   )
```

Parâmetros

index

[in] O índice do item alterado.

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Redraw

Redesenha o controle.

```
bool  Redraw()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

RowState

Define o estado do item especificado.

```
bool  RowState(
   const int   index,       // item index
   const bool  select       // state
   )
```

Parâmetros

index

[in] Índice do item para alterar.

select

[in] Novo estado.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Select

Seleciona item atual.

```
void  Select(
   const int  index      // index
   )
```

Parâmetros

index

[in] Índice do item para selecionar.

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CSpinEdit

CSpinEdit é uma classe de controle complexa (com controles dependentes).

Descrição

A classe CSpinEdit destina-se a criação de controles que permite editar os valores de tipo inteiro. Aumenta automaticamente os dados quando se pressiona o botão superior, mas diminui pressionando o botão inferior.

Declaração

```
class CSpinEdit : public CWndContainer
```

Título

```
#include <Controls\SpinEdit.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CSpinEdit

O resultado de trabalho é apresentado abaixo do código:

Métodos de classe

© 2000-2024, MetaQuotes Ltd.

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
MinValue	Obtém/Define o valor mínimo permitido
MaxValue	Obtém/Define o valor máximo permitido
State	
Value	Obtém/Define o valor atual
Controles dependentes	
CreateEdit	Cria um controle dependente (editar)
CreateInc	Cria controle dependente (botão de aumentar)
CreateDec	Cria controle dependente (botão de diminuir)
Manipuladores de controle de eventos dependentes	
OnClickInc	Manipulador de eventos "ClickInc"(virtual)
OnClickDec	Manipulador de eventos "ClickDec" (virtual)
Manipuladores de eventos internos	
OnChangeValue	Manipulador de eventos "ChangeValue" (virtual)

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Exemplo de criação de painel com rolagem de valores:

```
//+------------------------------------------------------------------+
```

© 2000-2024, MetaQuotes Ltd.

```
//|                                                ControlsSpinEdit.mq5 |
//|                          Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Painel de exibição e de diálogos de controle. Apresentação do t
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| defines                                                          |
//+------------------------------------------------------------------+
//--- indents and gaps
#define INDENT_LEFT                         (11)      // indent from left (with allowa
#define INDENT_TOP                          (11)      // indent from top (with allowar
#define INDENT_RIGHT                        (11)      // indent from right (with allov
#define INDENT_BOTTOM                       (11)      // indent from bottom (with allo
#define CONTROLS_GAP_X                      (5)       // gap by X coordinate
#define CONTROLS_GAP_Y                      (5)       // gap by Y coordinate
//--- for buttons
#define BUTTON_WIDTH                        (100)     // size by X coordinate
#define BUTTON_HEIGHT                       (20)      // size by Y coordinate
//--- for the indication area
#define EDIT_HEIGHT                         (20)      // size by Y coordinate
//--- for group controls
#define GROUP_WIDTH                         (150)     // size by X coordinate
#define LIST_HEIGHT                         (179)     // size by Y coordinate
#define RADIO_HEIGHT                        (56)      // size by Y coordinate
#define CHECK_HEIGHT                        (93)      // size by Y coordinate
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: main dialog of the Controls application                   |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                     // CSpinEdit objeto

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- create
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- chart event handler
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- create dependent controls
```

```
   bool              CreateSpinEdit(void);
   //--- handlers of the dependent controls events
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Event Handling                                                   |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Create                                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- create dependent controls
   if(!CreateSpinEdit())
      return(false);
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Create the "SpinEdit" element                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- coordinates
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- create
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_spin_edit))
```

© 2000-2024, MetaQuotes Ltd.

```
      return(false);
   m_spin_edit.MinValue(10);
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
//--- succeed
   return(true);
  }
//+------------------------------------------------------------------+
//| Event handler                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- create application dialog
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- run application
   ExtDialog.Run();
//--- succeed
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- limpamos os comentários
   Comment("");
//--- destroy dialog
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // event ID
                  const long& lparam,   // event parameter of the long type
                  const double& dparam, // event parameter of the double type
                  const string& sparam) // event parameter of the string type
```

© 2000-2024, MetaQuotes Ltd.

```
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CSpinEdit.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

MinValue (Método Get)

Obtém o valor da propriedade "MinValue" (valor mínimo permitido) do controle.

```
int  MinValue()  const
```

Valor de retorno

O valor da propriedade "MinValue".

MinValue (Método Set)

Define o valor da propriedade "MinValue" (valor mínimo permitido) do controle.

```
void  MinValue(
   const int  value      // new value
   )
```

Parâmetros

value

[in] O novo valor da propriedade "MinValue".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

MaxValue (Método Get)

Obtém o valor da propriedade "MaxValue" (valor máximo permitido) do controle.

```
int  MaxValue()  const
```

Valor de retorno

O valor da propriedade "MaxValue".

MaxValue (Método Set)

Define o o valor da propriedade "MaxValue" (valor máximo permitido) do controle.

```
void  MaxValue(
   const int  value      // new value
   )
```

Parâmetros

value

[in] Novo valor da propriedade "MaxValue".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Value (Método Get)

Obtém a propriedade "Value" (valor atual) do controle.

```
int  Value()  const
```

Valor de retorno

The "Value" property.

Value (Método Set)

Define a propriedade "Value" (valor atual) do controle.

```
void  Value(
   const int  value      // value
   )
```

Parâmetros

value

[in] Nova propriedade "Value"".

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

CreateEdit

Cria controle dependente (CEdit).

```
virtual bool  CreateEdit()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateInc

Cria controle dependente (botão de aumentar).

```
virtual bool  CreateInc()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateDec

Cria controle dependente (botão de diminuir).

```
virtual bool  CreateDec()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickInc

O manipulador de eventos do controle "ClickInc" (clique do mouse sobre o botão de aumentar).

```
virtual bool  OnClickInc()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnClickDec

O manipulador de eventos do controle "ClickDec" (clique do mouse sobre o botão de diminuir).

```
virtual bool  OnClickDec()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

OnChangeValue

O manipulador de eventos do controle "ChangeValue" .

```
virtual bool  OnChangeValue()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

CDialog

CDialog é de classe de controle complexa.

Descrição

CDialog destina-se a combinar os controles com funções diferentes no grupo.

Declaração

```
class CDialog : public CWndContainer
```

Título

```
#include <Controls\Dialog.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CDialog

Descendentes diretos

CAppDialog

Métodos de classe

Create	
Create	Criar controle
Manipuladores de eventos do gráfico	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Propriedades	
Caption	Obtém/Define o valor da propriedade "Caption"
Add	
Add	Adiciona controle para a área de cliente
Controles dependentes	
CreateWhiteBorder	Cria controle dependente (borda branca)
CreateBackground	Cria controle dependente (segundo plano-fundo)
CreateCaption	Cria controle dependente (subtítulo)
CreateButtonClose	Cria controle dependente (botão fechar)

© 2000-2024, MetaQuotes Ltd.

Create	
CreateClientArea	Cria controle dependente (área de cliente)
Manipuladores de controle de eventos dependentes	
OnClickCaption	Manipulador de evento "ClickCaption"
OnClickButtonClose	Manipulador de evento"ClickButtonClose"
Acesso à área de cliente	
ClientAreaVisible	Define um valor indicando se a área do cliente está visível
ClientAreaLeft	Obtém coordenada X do canto superior esquerdo do controle da área de cliente
ClientAreaTop	Obtém a coordenada Y do canto superior esquerdo do controle da área de cliente
ClientAreaRight	Obtém coordenadas X do canto inferior direito do controle da área de cliente
ClientAreaBottom	Obtém a coordenada Y do canto inferior direito do controle da área de cliente
ClientAreaWidth	Obtém a largura da área de cliente
ClientAreaHeight	Obtém a altura área de cliente
Manipulador de eventos para arrastar	
OnDialogDragStart	Manipulador de eventos "DialogDragStart" (virtual)
OnDialogDragProcess	Manipulador de eventos "DialogDragProcess" (virtual)
OnDialogDragEnd	Manipulador de eventos "DialogDragEnd" (virtual)

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CDialog.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Caption (Método Get)

Obtém a propriedade "Caption" do controle CDialog.

```
string  MinValue()  const
```

Valor de retorno

A propriedade"Caption".

Caption (Método Set)

Define a propriedade "Caption" do controle CDialog.

```
bool  Caption(
   const string  text      // text
   )
```

Parâmetros

text

[in] O novo valor da propriedade "Caption".

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Add

Adiciona controle para a área do cliente pelo ponteiro.

```
bool  Add(
   CWnd   *control,         // pointer
   )
```

Parâmetros

control

[in] Ponteiro para controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

Add

Adiciona controle para a área do cliente por referência.

```
bool  Add(
   CWnd   &control,         // reference
   )
```

Parâmetros

control

[in] Referência para controle.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateWhiteBorder

Cria controle dependente (borda branca).

```
virtual bool  CreateWhiteBorder()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateBackground

Cria controle dependente (segundo plano - fundo).

```
virtual bool  CreateBackground()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateCaption

Cria controle dependente (subtítulo).

```
virtual bool  CreateCaption()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateButtonClose

Cria controle dependente (botão fechar)

```
virtual bool  CreateButtonClose()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateClientArea

Cria controle dependente (área de cliente).

```
virtual bool  CreateClientArea()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickCaption

O manipulador de eventos do controle "ClickCaption".

```
virtual bool  OnClickCaption()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

OnClickButtonClose

O manipulador de eventos do controle "ClickButtonClose".

```
virtual bool  OnClickButtonClose()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ClientAreaVisible

Define um flag que indica se a área do cliente é visível.

```
bool  ClientAreaVisible(
   const bool  visible      // visibility flag
   )
```

Parâmetros

visible

[in] Flag visível.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ClientAreaLeft

Obtém Coordenada X do canto superior esquerdo do controle da área de cliente.

```
int  ClientAreaLeft()
```

Valor de retorno

A coordenada X do canto superior esquerdo do controle da área de cliente.

© 2000-2024, MetaQuotes Ltd.

ClientAreaTop

Obtém Coordenada Y do canto superior esquerdo do controle da área de cliente.

```
int  ClientAreaTop()
```

Valor de retorno

As coordenadas Y do canto superior esquerdo do controle da área de cliente.

© 2000-2024, MetaQuotes Ltd.

ClientAreaRight

Obtém Coordenada X do canto inferior direito do controle da área de cliente.

```
int  ClientAreaTop()
```

Valor de retorno

A coordenada X do canto inferior direito do controle da área de cliente.

© 2000-2024, MetaQuotes Ltd.

ClientAreaBottom

Obtém Coordenada Y do canto inferior direito do controle da área de cliente.

```
int  ClientAreaBottom()
```

Valor de retorno

A coordenada Y do canto inferior direito do controle da área de cliente.

© 2000-2024, MetaQuotes Ltd.

ClientAreaWidth

Obtém a largura do controle da área de cliente.

```
int  ClientAreaWidth()
```

Valor de retorno

A largura da área de cliente.

© 2000-2024, MetaQuotes Ltd.

ClientAreaHeight

Obtém a altura do controle da área de cliente .

```
int  ClientAreaHeight()
```

Valor de retorno

A altura do controle da área de cliente .

© 2000-2024, MetaQuotes Ltd.

OnDialogDragStart

O manipulador de eventos do controle "DialogDragStart".

```
virtual bool  OnDialogDragStart()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "DialogDragStart" ocorre no início, quando o controle é arrastado.

© 2000-2024, MetaQuotes Ltd.

OnDialogDragProcess

O manipulador de eventos do controle "DialogDragProcess"

```
virtual bool  OnDialogDragProcess()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "DialogDragProcess" ocorre quando o controle é arrastado.

© 2000-2024, MetaQuotes Ltd.

OnDialogDragEnd

O manipulador de eventos do controle "DialogDragEnd".

```
virtual bool  OnDialogDragEnd()
```

Valor de retorno

verdadeiro se o evento foi processado, caso contrário será falso.

Observação

O evento "DialogDragEnd" ocorre no final e o controle é arrastado.

© 2000-2024, MetaQuotes Ltd.

CAppDialog

CAppDialog é uma classe de Diálogo de Aplicativo de controle complexo (com controles dependentes).

Descrição

Classe CAppDialog destina-se a combinar os controles com diferentes funções no grupo dentro do programa MQL5.

Declaração

```
class CAppDialog : public CDialog
```

Título

```
#include <Controls\Dialog.mqh>
```

Hierarquia de herança

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

Métodos de classe

Criar e destruir	
Create	Criar controle
Destroy	Destruir controle
Processamento de eventos	
OnEvent	Manipulador de eventos de todos os eventos de gráfico
Run	
Run	Executa controle
Processamento de eventos de gráfico	
ChartEvent	Manipulador de eventos de todos os eventos de gráfico
Configuração	
Minimized	Define um valor indicando se o controle é minimizado
Save/Load	
IniFileSave	Salva o estado de controle para o arquivo
IniFileLoad	Carrega o estado de controle a partir do arquivo

© 2000-2024, MetaQuotes Ltd.

Criar e destruir	
IniFileName	Define o nome do arquivo para carregar/salvar o estado de controle
IniFileExt	Define a extensão do arquivo para carregar/salvar o estado de controle
Inicialização	
CreateCommon	Método de inicialização comum
CreateExpert	Método de inicialização para trabalhar em Expert Advisors
CreateIndicator	Método de inicialização para trabalhar em indicadores
Controles dependentes	
CreateButtonMinMax	Cria controles dependentes (minimizar/maximizar botões)
Manipuladores de controle de eventos dependentes	
OnClickButtonClose	Manipulador de eventos"ClickButtonClose" (virtual)
OnClickButtonMinMax	Manipulador de eventos "ClickButtonMinMax" (virtual)
Eventos externos	
OnAnotherApplicationClose	Manipulador de eventos externos (virtual)
Métodos	
Rebound	Define novas coordenadas do controle usando coordenadas de classe CRect
Minimizar	Mostra o controle no estado minimizado
Maximizar	Mostra o controle no estado maximizado (restaurado)
CreateInstanceId	Cria um ID único para os nomes dos objetos de controle
ProgramName	Obtém o nome do programa MQL5
SubwinOff	Obtém o deslocamento Y da sub-janela de controle

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

Métodos herdados da classe CWnd

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Métodos herdados da classe CWndContainer

© 2000-2024, MetaQuotes Ltd.

Métodos herdados da classe CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Métodos herdados da classe CDialog

Caption, Caption, Add, Add

© 2000-2024, MetaQuotes Ltd.

Create

Cria novo controle CAppDialog.

```
virtual bool  Create(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Destroy

Método de desinicialização do controle CAppDialog.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // código de motivo
)
```

Parâmetros

reason

[in] Código de motivo da desinicialização. Por padrão, é definido REASON_PROGRAM.

Valor retornado

Não.

© 2000-2024, MetaQuotes Ltd.

OnEvent

Manipulador de eventos do gráfico.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Run

Executa o controle.

```
bool  Run()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

ChartEvent

Manipulador de eventos do gráfico.

```
virtual bool  ChartEvent(
   const int      id,          // ID
   const long&    lparam,      // event parameter of long type
   const double&  dparam,      // event parameter of double type
   const string&  sparam       // event parameter of string type
   )
```

Parâmetros

id

[in] ID do evento.

lparam

[in] Parâmetro do evento tipo long, passado por referência.

dparam

[in] Parâmetro do evento tipo double, passado por referência

sparam

[in] Parâmetro do evento tipo string, passado por referência

Valor de retorno

verdadeiro - se o evento foi processado, caso contrário será falso.

© 2000-2024, MetaQuotes Ltd.

Minimized

Define o valor da propriedade do controle "Minimized" .

```
bool  Minimized(
   const bool  flag      // state
   )
```

Parâmetros

flag

[in] Novo estado.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

IniFileSave

Salva o estado de controle para o arquivo.

```
void  IniFileSave()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

IniFileLoad

Carrega o estado de controle a partir do arquivo.

```
void  IniFileLoad()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

IniFileName

Define o nome do arquivo para carregar/salvar o estado de controle.

```
virtual string  IniFileName()  const
```

Valor de retorno

O nome do arquivo para carregar/salvar o estado de controle.

Observação

O nome do arquivo incluem o nome do Indicador/Expert Advisor e o símbolo de trabalho, onde qual o programa é lançado no MQL5.

© 2000-2024, MetaQuotes Ltd.

IniFileExt

Define a extensão do arquivo para carregar/salvar o estado do controle.

```
virtual string  IniFileExt()  const
```

Valor de retorno

Extensão do arquivo, usado para carregar/salvar o estado de controle.

© 2000-2024, MetaQuotes Ltd.

CreateCommon

Método de inicialização comum.

```
bool  CreateCommon(
   const long    chart,       // chart ID
   const string  name,        // name
   const int     subwin,      // chart subwindow
   )
```

Parâmetros

chart

[in] ID de gráfico.

name

[in] Nome exclusivo do controle.

subwin

[in] Sub-janela do gráfico.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateExpert

Método de inicialização para trabalhar em Expert Advisors.

```
bool  CreateExpert(
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateIndicator

Método de inicialização para trabalhar em indicadores.

```
bool  CreateIndicator(
   const int     x1,          // x1 coordinate
   const int     y1,          // y1 coordinate
   const int     x2,          // x2 coordinate
   const int     y2           // y2 coordinate
   )
```

Parâmetros

x1

[in] Coordenada X do canto superior esquerdo.

y1

[in] Coordenada Y do canto superior esquerdo.

x2

[in] Coordenada X do canto inferior direito.

y2

[in] Coordenada Y do canto inferior direito.

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateButtonMinMax

Cria controles dependentes (botões minimizar/maximizar).

```
virtual void  CreateButtonMinMax()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnClickButtonClose

O controle do manipulador de eventos "ClickButtonClose" (clique do mouse sobre o botão para fechar).

```
virtual void  OnClickButtonClose()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnClickButtonMinMax

O controle do manipulador de eventos "ClickButtonMinMax" (clique do mouse sobre botão para minimizar/maximizar).

```
virtual void  OnClickButtonClose()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

OnAnotherApplicationClose

Manipulador de eventos para eventos externos.

```
virtual void  OnAnotherApplicationClose()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Rebound

Define novas coordenadas do controle usando as coordenadas da classe CRect.

```
bool  Rebound(
   const & CRect  rect      // CRect class
   )
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Minimizar

Mostra o controle no estado minimizado.

```
virtual void  Minimize()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

Maximizar

Mostra o controle no estado maximizado (restaurado).

```
virtual void  Maximize()
```

Valor de retorno

verdadeiro se obteve êxito, caso contrário falso.

© 2000-2024, MetaQuotes Ltd.

CreateInstanceId

Cria uma ID único para os nomes dos objetos de controle.

```
string  CreateInstanceId()
```

Valor de retorno

Prefixo para os nomes de objeto.

© 2000-2024, MetaQuotes Ltd.

ProgramName

Obtém o nome do programa MQL5.

```
string  ProgramName()
```

Valor de retorno

Nome do programa MQL5.

© 2000-2024, MetaQuotes Ltd.

SubwinOff

Obtém a deslocamento Y da sub-janela de controle.

```
void  SubwinOff()
```

Valor de retorno

Nenhum.

© 2000-2024, MetaQuotes Ltd.

Migrando da linguagem de programação MQL4 para MQL5

MQL5 é a evolução do seu antecessor - a linguagem de programação MQL4, onde inúmeros indicadores, scripts e Expert Advisors foram escritos. Apesar do fato de que a nova linguagem de programação é grandemente compatível com a linguagem da geração anterior, ainda existem algumas diferenças entre as linguagens. E quando migrar de programas, essas diferenças devem ser observadas.

Esta seção contém informações destinadas a facilitar a adaptação dos códigos para a nova linguagem MQL5 ao programadores que conhecem MQL4.

Em primeiro lugar, deve-se observar:

- A nova linguagem não contém funções start(), init() e deinit().
- Não há limite para o número de buffers de indicador.
- DLLs são carregadas imediatamente após o carregamento de um Expert Advisor (ou qualquer outro programa MQL5).
- Verificação das condições lógicas é reduzida.
- Quando os limites de um matriz forem ultrapassados, o desempenho em curso é finalizado (criticamente - com a saída de um erro).
- Originários de operadores em C++.
- A linguagem oferece o tipo de conversão implícita (mesmo de string para um número).
- As variáveis locais não são inicializados automaticamente (exceto para strings).
- Matrizes de local comum são excluídas automaticamente.

Funções Especiais init, start e deinit

A linguagem MQL4 contém apenas estas três funções pré-definidas que são usadas no indicador, script ou Expert Advisor (não contando a inclusão dos arquivos *.mqh e dos arquivos de biblioteca). Em MQL5 não existem estas funções, mas foram criadas funções análogas. A tabela mostra a correspondência aproximada das funções.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

Funções OnInit e OnDeinit desempenham o mesmo papel que as funções init e deinit em MQL4 - elas são projetadas para encontrar o código, que será realizado durante a inicialização e finalização dos programas MQL5. Você pode simplesmente renomear estas funções em conformidade, ou deixá-las como estão, mas deve adicionar chamadas dessas funções em lugares correspondentes.

Exemplo:

```
void OnInit()
  {
//--- Função nomeada durante a inicialização
```

© 2000-2024, MetaQuotes Ltd.

```
   init();
  }
void OnDeinit(const int reason)
  {
//--- Função nomeada durante a desinicialização
   deinit();
//---
  }
```

A função start é substituída por OnStart apenas em scripts. Em Expert Advisors e indicadores será renomeado para OnTick e OnCalculate, respectivamente. O código que está para ser executado durante a operação do programa MQL5 será centrado nestas três funções:

programa mql5	principal função
script	OnStart
indicador	OnCalculate
Expert Advisor	OnTick

Se o código do indicador ou script não contém a função principal, ou o nome da função é diferente, esta função não será executada. Isso significa que se o código fonte de um script não contém OnStart, esse código será compilado como um Expert Advisor.

Se um código de indicador não contém a função OnCalculate, a compilação deste indicador será impossível.

Variáveis Predefinidas

Em MQL5 não existem as variáveis pré-definidas como Ask, Bid, Bars. As variáveis de Point e Digits têm uma grafia ligeiramente diferente:

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

Acesso as Timeseries

© 2000-2024, MetaQuotes Ltd.

Em MQL5 não existem as timeseries predefinidas como Open[], High[], Low[], Close[], Volume[] e Time[]. A profundidade necessária de uma timeseries agora pode ser definida através das funções correspondentes para acessar as timeseries.

Expert Advisors

Expert Advisors em MQL5 não requerem a presença obrigatória de funções que manipulam os eventos de recebimento de um novo tick - OnTick, com é realizado em MQL4 (a função de arranque em MQL4 é executado quando um novo tick é recebido). Em MQL5, os Expert Advisors podem conter funções pré-definidas do manipulador de vários tipos de eventos:

- OnTick – recebimento de um novo tick;
- OnTimer – evento de timer;
- OnTrade - evento de trade;
- OnChartEvent – eventos de entrada do teclado e mouse, eventos de um objeto gráfico em movimento, evento de edição de texto completo na entrada da etiqueta do objeto LabelEdit;
- OnBookEvent – evento do status de mudança da Profundidade do Mercado.

Indicadores Customizados

Em MQL4, o número de buffers de indicador é limitado e não pode ultrapassar a 8. Em MQL5 não existem estas limitações, mas deve ser lembrado que cada buffer de indicador requer a alocação de uma certa parte da memória para a sua localização no terminal, por isto esta nova possibilidade não deve ser utilizada abusivamente.

MQL4 oferece apenas 6 tipos de plotagem de indicadores personalizados, enquanto MQL5 agora oferece 18 estilos de desenho. Os nomes dos tipos de desenho não mudaram, mas a ideologia da representação gráfica dos indicadores mudou significativamente.

A direção da indexação nos buffers de indicador também é diferente. Por padrão, em MQL5 todos os buffers de indicador tem o comportamento de matrizes comuns, ou seja, o elemento 0 (zero) indexado é o mais antigo no histórico e, como os índices aumentam, passamos dos dados mais antigos para os mais recentes.

A única função para trabalhar com indicadores personalizados que foi preservado da linguagem MQL4 é a função SetIndexBuffer, mas para ser utilizada; agora deve ser especificado o tipo de dado a ser armazenado em um matriz, ligada ao buffer do indicador.

As propriedades de indicadores personalizados também foram modificadas e ampliadas. Novas funções para acesso a timeseries foram adicionadas, de modo que o algoritmo de cálculo total deve ser reexaminado.

Graphical Objects

O número de objetos gráficos em MQL5 foi aumentado significativamente. Além disso, objetos gráficos agora são posicionados no gráfico com a precisão de um segundo em qualquer período de tempo - agora objetos de pontos de ancoragem não são arredondados para o tempo de abertura da barra de preço no gráfico em curso.

© 2000-2024, MetaQuotes Ltd.

Agora para Setas, Textos e Objetos de Etiqueta você pode especificar métodos de vinculação, e para a Etiqueta, Botão, Gráfico, Etiqueta Bitmap e Editar, os objetos podem ser configurados no canto do gráfico onde um objeto está ligado.

© 2000-2024, MetaQuotes Ltd.

Lista de Funções MQL5

Todas as funções MQL5 em ordem alfabética.

Função	Ação	Section
AccountInfoDouble	Retorna um valor de tipo double da correspondente propriedade da conta	Informações de Conta
AccountInfoInteger	Retorna um valor de tipo inteiro (bool, int ou long) da correspondente propriedade da conta	Informações de Conta
AccountInfoString	Retorna um valor de tipo string da correspondente propriedade da conta	Informações de Conta
acos	Retorna o arco-cosseno de x em radianos	Funções Matemáticas
Alert	Exibe uma mensagem em uma janela separada	Funções Comuns
ArrayBsearch	Searches for a specified value in a multidimensional numeric array sorted ascending	Funções para Array
ArrayCompare	Retorna o resultado da comparação entre dois arrays de tipos simples ou estruturas personalizados sem objetos complexos	Funções para Array
ArrayCopy	Copia um array em outro array	Funções para Array
ArrayFill	Preenche um array com o valor especificado	Funções para Array
ArrayFree	Libera buffer de qualquer array dinâmico e define o tamanho da dimensão zero em 0.	Funções para Array
ArrayGetAsSeries	Verifica a direção da indexação de um array	Funções para Array
ArrayInitialize	Define todos os elementos de um array numérico para um único valor	Funções para Array
ArrayIsDynamic	Verifica se um array é dinâmico	Funções para Array
ArrayIsSeries	Verifica se um é array é uma série de tempo	Funções para Array

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
ArrayMaximum	Searches for the largest element in the first dimension of a multidimensional numeric array	Funções para Array
ArrayMinimum	Searches for the lowest element in the first dimension of a multidimensional numeric array	Funções para Array
ArrayRange	Retorna o número de elementos na dimensão especifica do array	Funções para Array
ArrayResize	Define o novo tamanho da primeira dimensão do array	Funções para Array
ArraySetAsSeries	Define a direção de indexação de um array	Funções para Array
ArraySize	Retorna o número de elemento no array	Funções para Array
ArraySort	Ordena arrays números pela primeira dimensão	Funções para Array
ArrayPrint	No log, exibe uma matriz de tipo ou estrutura simples	Funções para Array
ArrayInsert	Insere na matriz de destino o número especificado de elementos da matriz de origem, iniciando no índice especificado	Funções para Array
ArrayRemove	Remove o número especificado de elementos da matriz, iniciando no índice especificado	Funções para Array
ArrayReverse	Expande o número de elementos especificado na matriz, iniciando no índice especificado	Funções para Array
ArraySwap	Troca entre si o conteúdo de duas matrizes do tipo dinâmico	Funções para Array
asin	Retorna o arco-seno de x em radianos	Funções Matemáticas
atan	Retorna o arco-tangente de x em radianos	Funções Matemáticas
Bars	Retorna o número de barras do histórico para um ativo e período especificados	Séries Temporais e Acesso a Indicadores
BarsCalculated	Retorna o número de dados calculados em um buffer de	Séries Temporais e Acesso a Indicadores

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	indicador ou -1 em caso de erro (os dados não foram calculados ainda)	
CalendarCountryById	Obtém uma descrição do país por seu ID	Calendário econômico
CalendarEventById	Obtém a descrição do evento por seu ID	Calendário econômico
CalendarValueById	Obtém uma descrição do valor do evento por seu ID	Calendário econômico
CalendarCountries	Obtém um array de descrições dos países disponíveis no calendário.	Calendário econômico
CalendarEventByCountry	Obtém um array de descrições de todos os eventos disponíveis no Calendário, segundo o código do país especificado	Calendário econômico
CalendarEventByCurrency	Obtém um array de descrições de todos os eventos disponíveis no calendário, de acordo com a moeda especificada	Calendário econômico
CalendarValueHistoryByEvent	Obtém um array de valores de todos os eventos no intervalo de tempo especificado, segundo o ID do evento	Calendário econômico
CalendarValueHistory	Obtém uma matriz de valores de todos os eventos num determinado intervalo de tempo com um filtro por país e/ou moeda	Calendário econômico
CalendarValueLastByEvent	Obtém um array de valores do eventos por seu ID desde o momento em que criado do banco de dados do Calendário com o change_id definido	Calendário econômico
CalendarValueLast	Obtém um array de valores de todos os eventos com filtro por país e/ou moeda desde o momento em que criado do banco de dados do Calendário com o change_id definido	Calendário econômico
ceil	Retorna o valor numérico inteiro mais próximo do acima	Funções Matemáticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
CharArrayToString	Converte código de símbolo (ansi) em um array de um símbolo	Funções de Conversão
ChartApplyTemplate	Aplica-se um modelo específico de um arquivo especificado para o gráfico	Operações de Gráficos
ChartClose	Fecha um gráfico especificado	Operações de Gráficos
ChartFirst	Retorna o ID do primeiro gráfico do terminal do cliente	Operações de Gráficos
ChartGetDouble	Retorna a propriedade de valor duplo do gráfico especificado	Operações de Gráficos
ChartGetInteger	Retorna a propriedade de valor inteiro do gráfico especificado	Operações de Gráficos
ChartGetString	Retorna a propriedade de valor da seqüência do gráfico especificado	Operações de Gráficos
ChartID	Retorna o ID do gráfico atual	Operações de Gráficos
ChartIndicatorAdd	Adiciona um indicador com identificador especificado para uma janela do gráfico especificado	Operações de Gráficos
ChartIndicatorDelete	Remove um indicador com um nome especificado a partir de janela do gráfico especificado	Operações de Gráficos
ChartIndicatorGet	Retorna o manuseio do indicador com nome abreviado especificado na janela do gráfico especificado	Operações de Gráficos
ChartIndicatorName	Retorna o nome abreviado do indicador pelo número na lista de indicadores sobre a janela do gráfico especificado	Operações de Gráficos
ChartIndicatorsTotal	Retorna o número de todos indicadores utilizados para a janela do gráfico especificado	Operações de Gráficos
ChartNavigate	Executa deslocamento do gráfico especificado pelo número especificado de barras em relação a posição especificada no gráfico	Operações de Gráficos
ChartNext	Retorna o ID do gráfico do próximo gráfico para um especificado	Operações de Gráficos

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
ChartOpen	Abre um novo gráfico com o símbolo e período especificado	Operações de Gráficos
CharToString	Converte um código de símbolo em um string de um caractere.	Funções de Conversão
ChartPeriod	Retorna o valor do período do gráfico especificado	Operações de Gráficos
ChartPriceOnDropped	Retorna a coordenada de preço do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado	Operações de Gráficos
ChartRedraw	Chama um redesenho forçado de um gráfico especificado	Operações de Gráficos
ChartSaveTemplate	Salva as configurações atuais do gráfico em um modelo com um nome especificado	Operações de Gráficos
ChartScreenShot	Providencia uma imagem do gráfico de seu estado atual para um formato GIF, PNG ou BMP dependendo da extensão especificada	Operações de Gráficos
ChartSetDouble	Define o valor duplo para uma propriedade correspondente do gráfico especificado	Operações de Gráficos
ChartSetInteger	Define um valor inteiro (datetime, int, color, bool ou char) para uma propriedade correspondente do gráfico especificado	Operações de Gráficos
ChartSetString	Define o valor da seqüência para uma propriedade do gráfico especificado	Operações de Gráficos
ChartSetSymbolPeriod	Modifica o valor do símbolo e um período do gráfico especificado	Operações de Gráficos
ChartSymbol	Retorna o nome do símbolo do gráfico especificado	Operações de Gráficos
ChartTimeOnDropped	Retorna a coordenada de tempo do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado	Operações de Gráficos
ChartTimePriceToXY	Converte as coordenadas do gráfico a partir da representação	Operações de Gráficos

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	tempo/preço para as coordenadas X e Y	
ChartWindowFind	Retorna o número de uma sub-janela onde um indicador é elaborado	Operações de Gráficos
ChartWindowOnDropped	Retorna o número (índice) da sub-janela do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado	Operações de Gráficos
ChartXOnDropped	Retorna a coordenada X do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado	Operações de Gráficos
ChartXYToTimePrice	Converte as coordenadas X e Y no gráfico para os valores de tempo e preço	Operações de Gráficos
ChartYOnDropped	Retorna a coordenada Y do ponto do gráfico, o Assessor Especialista (Expert Advisor) ou Script foi derrubado	Operações de Gráficos
CheckPointer	Retorna o tipo do ponteiro de objeto	Funções Comuns
CLBufferCreate	Cria um buffer OpenCL	Trabalhando com OpenCL
CLBufferFree	Exclui um buffer OpenCL	Trabalhando com OpenCL
CLBufferRead	Lê um buffer de OpenCL num array	Trabalhando com OpenCL
CLBufferWrite	Grava um array num buffer OpenCL	Trabalhando com OpenCL
CLContextCreate	Cria um contexto OpenCL	Trabalhando com OpenCL
CLContextFree	Remove um contexto OpenCL	Trabalhando com OpenCL
CLExecute	Executa um programa OpenCL	Trabalhando com OpenCL
CLGetDeviceInfo	Receives device property from OpenCL driver	Trabalhando com OpenCL
CLGetInfoInteger	Retorna o valor da propriedade de uma integer para um objeto ou dispositivo OpenCL	Trabalhando com OpenCL
CLHandleType	Retorna o tipo do manipulador de OpenCL como valor da	Trabalhando com OpenCL

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	enumeração ENUM_OPENCL_HANDLE_TYPE	
CLKernelCreate	Cria uma função de partida OpenCL	Trabalhando com OpenCL
CLKernelFree	Remove uma função de partida OpenCL	Trabalhando com OpenCL
CLProgramCreate	Cria um programa OpenCL a partir de um código fonte	Trabalhando com OpenCL
CLProgramFree	Remove um programa OpenCL	Trabalhando com OpenCL
CLSetKernelArg	Define um parâmetro para a função OpenCL	Trabalhando com OpenCL
CLSetKernelArgMem	Define um buffer OpenCL como um parâmetro da função OpenCL	Trabalhando com OpenCL
ColorToARGB	Converte tipo color para tipo uint para receber representação ARGB da cor.	Funções de Conversão
ColorToString	Converte um valor de color em uma string como "R,G,B"	Funções de Conversão
Comment	Gera um comentário no canto superior esquerdo do gráfico	Funções Comuns
CopyBuffer	Copia dados de um buffer especificado de um indicador especificado em um array	Séries Temporais e Acesso a Indicadores
CopyClose	Obtém histórico de dados sobre preço de fechamento de barra para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyHigh	Obtém histórico de dados sobre preço de barra máximo para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyLow	Obtém histórico de dados sobre preço de barra minimo para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyOpen	Obtém dados históricos sobre preço de abertura de barras para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
CopyRates	Obtém dados históricos da estrutura Rates para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyRealVolume	Obtém dados históricos sobre volumes de negociação para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopySpread	Obtém histórico de dados sobre spreads para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyTicks	Gets ticks accumulated by the terminal for the current working session into an array	Séries Temporais e Acesso a Indicadores
CopyTickVolume	Obtém histórico de dados sobre volumes de tick para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
CopyTime	Obtém dados históricos sobre hora de abertura de barras para um ativo e período especificados em um array	Séries Temporais e Acesso a Indicadores
cos	Retorna o cosseno de um número	Funções Matemáticas
CryptDecode	Performs the inverse transformation of the data from array	Funções Comuns
CryptEncode	Transforms the data from array with the specified method	Funções Comuns
CustomSymbolCreate	Cria um símbolo personalizado com o nome especificado no grupo definido	Símbolos personalizados
CustomSymbolDelete	Remove o símbolo personalizado com o nome especificado	Símbolos personalizados
CustomSymbolSetInteger	Define o valor do tipo inteiro para o símbolo personalizado	Símbolos personalizados
CustomSymbolSetDouble	Define o valor do tipo real para o símbolo personalizado	Símbolos personalizados
CustomSymbolSetString	Define o valor do tipo cadeia de caracteres para o símbolo personalizado	Símbolos personalizados

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
CustomSymbolSetMarginRate	Define o coeficiente de cobrança da margem - dependendo do tipo e direção da ordem - para o símbolo personalizado	Símbolos personalizados
CustomSymbolSetSessionQuote	Define a hora de início e fim da sessão de cotação especificada para o símbolo especificado e dia da semana	Símbolos personalizados
CustomSymbolSetSessionTrade	Define a hora de início e fim da sessão de negociação especificada para o símbolo especificado e dia da semana	Símbolos personalizados
CustomRatesDelete	Exclui todas as barras no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado	Símbolos personalizados
CustomRatesReplace	Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlRates, no intervalo de tempo definido	Símbolos personalizados
CustomRatesUpdate	Adiciona ao histórico do instrumento personalizado as barras que faltam e substitui os dados existentes na matriz do tipo MqlRates	Símbolos personalizados
CustomTicksAdd	Adds data from an array of the MqlTick type to the price history of a custom symbol. The custom symbol must be selected in the Market Watch window.	Símbolos personalizados
CustomTicksDelete	Exclui todos os ticks no histórico de preço do instrumento personalizado, no intervalo de tempo selecionado	Símbolos personalizados
CustomTicksReplace	Substitui todo o histórico de preço do instrumento personalizado pelos dados na matriz do tipo MqlTick, no intervalo de tempo definido	Símbolos personalizados
CustomBookAdd	Envia o estado do livro de ofertas do instrumento personalizado	Símbolos personalizados

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
DatabaseOpen	Abre ou cria um banco de dados no arquivo especificado	Trabalhar com bancos de dados
DatabaseClose	Fecha um banco de dados	Trabalhar com bancos de dados
DatabaseImport	Importa para uma tabela dados de um arquivo	Trabalhar com bancos de dados
DatabaseExport	Exporta uma tabela ou resultado sobre a execução da consulta SQL para um arquivo CSV	Trabalhar com bancos de dados
DatabasePrint	Exibe uma tabela ou resultado sobre a execução da consulta SQL no log do EA	Trabalhar com bancos de dados
DatabaseTableExists	Verifica a existência de tabela no banco de dados	Trabalhar com bancos de dados
DatabaseExecute	Executa uma consulta no banco de dados especificado	Trabalhar com bancos de dados
DatabasePrepare	Cria um identificador de consulta, que pode ser executado usando DatabaseRead()	Trabalhar com bancos de dados
DatabaseReset	Redefine a consulta para seu status inicial, como após uma chamada de DatabasePrepare()	Trabalhar com bancos de dados
DatabaseBind	Define o valor do parâmetro na consulta	Trabalhar com bancos de dados
DatabaseBindArray	Define a matriz como um valor de parâmetro	Trabalhar com bancos de dados
DatabaseRead	Realiza o a transição para o próximo registro como resultado de uma consulta	Trabalhar com bancos de dados
DatabaseReadBind	Moves to the next record and reads data into the structure from it	Trabalhar com bancos de dados
DatabaseFinalize	Exclui uma consulta criada em DatabasePrepare()	Trabalhar com bancos de dados
DatabaseTransactionBegin	Inicia a execução da transação	Trabalhar com bancos de dados
DatabaseTransactionCommit	Conclui a execução da transação	Trabalhar com bancos de dados

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
DatabaseTransactionRollback	Executa a reversão de transações	Trabalhar com bancos de dados
DatabaseColumnsCount	Obtém o número de campos na consulta	Trabalhar com bancos de dados
DatabaseColumnName	Obtém o nome do campo pelo número	Trabalhar com bancos de dados
DatabaseColumnType	Obter tipo de campo por número	Trabalhar com bancos de dados
DatabaseColumnSize	Obtém o tamanho do campo em bytes	Trabalhar com bancos de dados
DatabaseColumnText	Obtém o valor do campo do registro atual como uma string	Trabalhar com bancos de dados
DatabaseColumnInteger	Obtém um valor int do registro atual	Trabalhar com bancos de dados
DatabaseColumnLong	Obtém um valor long do registro atual	Trabalhar com bancos de dados
DatabaseColumnDouble	Obtém um valor double do registro atual	Trabalhar com bancos de dados
DatabaseColumnBlob	Obtém o valor do campo do registro atual como um array	Trabalhar com bancos de dados
DebugBreak	Ponto de parada de programa em depuração	Funções Comuns
Digits	Retorna o número de dígitos decimais determinando a precisão do valor do preço do ativo do gráfico corrente	Verificando Estado
DoubleToString	Converte um valor numérico para um linha de texto com a precisão especificada	Funções de Conversão
DXContextCreate	Cria um contexto gráfico para renderizar quadros do tamanho especificado	Trabalho com DirectX
DXContextSetSize	Altera o tamanho do quadro do contexto gráfico criado em DXContextCreate()	Trabalho com DirectX
DXContextSetSize	Obtém o tamanho do quadro do contexto gráfico criado em DXContextCreate().	Trabalho com DirectX

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
DXContextClearColors	Define, para o buffer de renderização, todos os pixeis na cor especificada	Trabalho com DirectX
DXContextClearDepth	Limpa o buffer de profundidade	Trabalho com DirectX
DXContextGetColors	Obtém, do contexto gráfico, uma imagem com o tamanho e o deslocamento especificados	Trabalho com DirectX
DXContextGetDepth	Obtém o buffer de profundidade do quadro renderizado	Trabalho com DirectX
DXBufferCreate	Cria um buffer com tipo especificado baseado em uma matriz de dados	Trabalho com DirectX
DXTextureCreate	Cria uma textura bidimensional, a partir de um retângulo com determinado tamanho cortado da imagem transferida	Trabalho com DirectX
DXInputCreate	Cria os parâmetros de entrada do sombreador	Trabalho com DirectX
DXInputSet	Define os parâmetros de entrada do sombreador	Trabalho com DirectX
DXShaderCreate	Cria um sombreador com tipo especificado	Trabalho com DirectX
DXShaderSetLayout	Define o layout do vértice para o sombreador de vértice	Trabalho com DirectX
DXShaderInputsSet	Define os parâmetros de entrada do sombreador	Trabalho com DirectX
DXShaderTexturesSet	Define a textura do sombreador.	Trabalho com DirectX
DXDraw	Desenha os vértices do buffer de vértice definido em DXBufferSet()	Trabalho com DirectX
DXDrawIndexed	Desenha as primitivas gráficas descritas pelo buffer de índice de DXBufferSet()	Trabalho com DirectX
DXPrimiveTopologySet	Define o tipo de primitivas para renderização usando DXDrawIndexed()	Trabalho com DirectX
DXBufferSet	Define o buffer para a renderização atual	Trabalho com DirectX
DXShaderSet	Define o sombreador para renderização	Trabalho com DirectX

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
DXHandleType	Retorna o tipo de identificador	Trabalho com DirectX
DXRelease	Libera o identificador	Trabalho com DirectX
EnumToString	Converte um valor de enumeração de qualquer tipo para string	Funções de Conversão
EventChartCustom	Gera um evento personalizado para o gráfico especificado	Trabalhando com Eventos
EventKillTimer	Interrompe a geração de eventos pelo timer no quadro atual	Trabalhando com Eventos
EventSetMillisecondTimer	Inicia gerador de eventos de timer de alta resolução com um período de menos de 1 segundo para o gráfico em curso	Trabalhando com Eventos
EventSetTimer	Inicia o gerador de eventos do time com a periodicidade especificada para o gráfico atual	Trabalhando com Eventos
exp	Retorna o exponencial de um número	Funções Matemáticas
ExpertRemove	Interrompe um Expert Advisor e o descarrega do gráfico	Funções Comuns
fabs	Retorna o valor absoluto (modulo) do valor numérico especificado	Funções Matemáticas
FileClose	Fecha um arquivo aberto previamente	Funções de Arquivo
FileCopy	Copia o arquivo original a partir de um local ou de uma pasta compartilhada em outro arquivo	Funções de Arquivo
FileDelete	Deleta um arquivo especificado	Funções de Arquivo
FileFindClose	Fecha o manipulador de busca	Funções de Arquivo
FileFindFirst	Começa a busca de arquivos em um diretório de acordo com o filtro especificado	Funções de Arquivo
FileFindNext	Continua a busca iniciada pela função FileFindFirst()	Funções de Arquivo
FileFlush	Escreve para um disco todos os dados remanescentes dentro da entrada/saída do arquivo buffer	Funções de Arquivo
FileGetInteger	Obtém uma propriedade inteira de um arquivo	Funções de Arquivo

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
FileIsEnding	Define o final do arquivo em processo de leitura	Funções de Arquivo
FileIsExist	Verifica a e existência de um arquivo	Funções de Arquivo
FileIsLineEnding	Define o final da linha do arquivo de texto em processo de leitura	Funções de Arquivo
FileMove	Move ou renomeia um arquivo	Funções de Arquivo
FileOpen	Abre um arquivo com o nome e bandeira especificada	Funções de Arquivo
FileReadArray	Lê arrays de qualquer tipo, exceto para a seqüência a partir do arquivo do tipo BIN	Funções de Arquivo
FileReadBool	Lê a partir de arquivo do tipo CSV uma seqüência a partir da posição atual até um delimitador (ou até o fim de uma linha de texto) e converte a seqüência de leitura para um valor do tipo bool	Funções de Arquivo
FileReadDatetime	Lê a partir do arquivo do tipo CSV uma seqüência de um dos formatos: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" or "HH:MM:SS" - e converte-os dentro de valor de datetime	Funções de Arquivo
FileReadDouble	Lê um valor double a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
FileReadFloat	Lê um valor float a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
FileReadInteger	Lê um valor int, short ou char a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
FileReadLong	Lê um valor do tipo long a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
FileReadNumber	Lê a partir de arquivo do tipo CSV uma seqüência a partir da posição atual até um delimitador (ou até o fim de uma linha de texto) e	Funções de Arquivo

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	converte a seqüência de leitura em um valor double	
FileReadString	Lê uma seqüência a partir da posição atual do ponteiro a partir de um arquivo	Funções de Arquivo
FileReadStruct	Lê o conteúdo de um arquivo binário em uma estrutura passada como um parâmetro, a partir da posição atual do ponteiro do arquivo	Funções de Arquivo
FileSeek	Move a posição do ponteiro de arquivo por um determinado número de bytes em relação à posição especificada	Funções de Arquivo
FileSize	Retorna o tamanho da abertura de um arquivo correspondente	Funções de Arquivo
FileTell	Retorna a posição atual do ponteiro de um arquivo da abertura de arquivo correspondente	Funções de Arquivo
FileWrite	Escreve dados para um arquivo de CSV ou tipo TXT	Funções de Arquivo
FileWriteArray	Escreve arrays de qualquer tipo exceto para seqüência em um arquivo do tipo BIN	Funções de Arquivo
FileWriteDouble	Escreve valor do tipo double a partir da posição atual de um ponteiro de arquivo em um arquivo binário	Funções de Arquivo
FileWriteFloat	Escreve o valor do tipo float a partir da posição atual de um ponteiro de arquivo em um arquivo binário	Funções de Arquivo
FileWriteInteger	Escreve o valor do tipo int a partir da posição atual de um ponteiro de arquivo em um arquivo binário	Funções de Arquivo
FileWriteLong	Escreve o valor do tipo long a partir da posição atual de um ponteiro de arquivo em um arquivo binário	Funções de Arquivo

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
FileWriteString	Escreve o valor de um parâmetro da string em um arquivo BIN ou TXT iniciando a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
FileWriteStruct	Escreve o conteúdo de uma estrutura passada como um parâmetro em um arquivo binário, começando a partir da posição atual do ponteiro de arquivo	Funções de Arquivo
floor	Retorna o valor numérico inteiro mais próximo do abaixo	Funções Matemáticas
fmax	Retorna o valor máximo entre dois valores numéricos	Funções Matemáticas
fmin	Retorna de valor mínimo entre dois valores numéricos	Funções Matemáticas
fmod	Retorna o resto real da divisão de dois números	Funções Matemáticas
FolderClean	Deleta todos os arquivos dentro da pasta especificada	Funções de Arquivo
FolderCreate	Cria uma pasta dentro do diretório de arquivo	Funções de Arquivo
FolderDelete	Remove um diretório selecionado. Se a pasta não estiver vazia, então ele não pode ser removido	Funções de Arquivo
FrameAdd	Adiciona um quadro com dados	Trabalhando com Resultados de Otimização
FrameFilter	Define a estrutura do filtro de leitura e move o ponteiro para o início	Trabalhando com Resultados de Otimização
FrameFirst	Move um ponteiro de leitura de quadro para o início e redefine previamente um conjunto de filtros	Trabalhando com Resultados de Otimização
FrameInputs	Recebe parâmetros de entrada, no qual o quadro é formado	Trabalhando com Resultados de Otimização
FrameNext	Lê um quadro e move o ponteiro para o próximo	Trabalhando com Resultados de Otimização
GetLastError	Retorna o último erro	Verificando Estado

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
GetPointer	Retorna o ponteiro de objeto	Funções Comuns
GetTickCount	Retorno o número de milissegundos decorridos desde que o sistema foi iniciado	Funções Comuns
GlobalVariableCheck	Verifica a existência de uma variável global com o nome especificado	Variáveis Globais do Terminal
GlobalVariableDel	Exclui uma variável global	Variáveis Globais do Terminal
GlobalVariableGet	Retorna o valor da variável global	Variáveis Globais do Terminal
GlobalVariableName	Retorna o nome de uma variável global, que é o número ordinal dentro da lista de variáveis globais	Variáveis Globais do Terminal
GlobalVariablesDeleteAll	Exclui as variáveis globais com especificação prefixada em seus nomes	Variáveis Globais do Terminal
GlobalVariableSet	Define o novo valor para uma variável global	Variáveis Globais do Terminal
GlobalVariableSetOnCondition	Define um novo valor de existência de variável global pela condição estabelecida	Variáveis Globais do Terminal
GlobalVariablesFlush	Salva forçadamente o conteúdo de todas as varáveis globais num disco	Variáveis Globais do Terminal
GlobalVariablesTotal	Retorna o número total de Variáveis Globais	Variáveis Globais do Terminal
GlobalVariableTemp	Define o novo valor para uma variável global, que existe somente na sessão geral do terminal	Variáveis Globais do Terminal
GlobalVariableTime	Retorna o tempo do último acesso para a variável global	Variáveis Globais do Terminal
HistoryDealGetDouble	Retorna a propriedade solicitada de um acordo no histórico (double)	Funções de Negociação (Trade)
HistoryDealGetInteger	Retorna a propriedade solicitada de um acordo no histórico (datetime ou int)	Funções de Negociação (Trade)

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
HistoryDealGetString	Retorna a propriedade solicitada de uma operação no histórico (string)	Funções de Negociação (Trade)
HistoryDealGetTicket	Devolve um bilhete de uma quantidade correspondente no histórico	Funções de Negociação (Trade)
HistoryDealSelect	Seleciona um negócio no histórico para posterior chamada através de funções apropriadas	Funções de Negociação (Trade)
HistoryDealsTotal	Retorna o número de ofertas no histórico	Funções de Negociação (Trade)
HistoryOrderGetDouble	Retorna a propriedade solicitada de uma ordem no histórico (double)	Funções de Negociação (Trade)
HistoryOrderGetInteger	Retorna a propriedade solicitada de uma ordem no histórico (datetime ou int)	Funções de Negociação (Trade)
HistoryOrderGetString	Retorna a propriedade solicitada de uma ordem no histórico (string)	Funções de Negociação (Trade)
HistoryOrderGetTicket	Retornar bilhete ordem de uma ordem correspondente no histórico	Funções de Negociação (Trade)
HistoryOrderSelect	Seleciona uma ordem no histórico para posteriormente trabalhar com ele	Funções de Negociação (Trade)
HistoryOrdersTotal	Retorna o número de pedidos no histórico	Funções de Negociação (Trade)
HistorySelect	Recupera o histórico de transações e ordens para o período do tempo de servidor especificado	Funções de Negociação (Trade)
HistorySelectByPosition	Pedido de histórico de acordo com um identificador de posição especificada.	Funções de Negociação (Trade)
iBars	Retorna o número de barras - no histórico - do símbolo e do período correspondentes	Séries Temporais e Acesso a Indicadores
iBarShift	Retorna o índice da barra na qual o tempo especificado coincide	Séries Temporais e Acesso a Indicadores

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
iClose	Retorna o valor do preço de fechamento da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iHigh	Retorna o valor do preço máximo da barra (especificado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iHighest	Retorna o índice do maior valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iLow	Retorna o valor do preço mínimo da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iLowest	Retorna o índice do menor valor encontrado (deslocamento relativo à barra atual) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iOpen	Retorna o valor do preço de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iTime	Retorna o valor do tempo de abertura da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iTickVolume	Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iRealVolume	Retorna o valor do volume real da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iVolume	Retorna o valor do volume de tick (especificado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iSpread	Retorna o valor do spread da barra (indicado pelo parâmetro shift) do gráfico correspondente	Séries Temporais e Acesso a Indicadores
iAD	Accumulation/Distribution	Indicadores Técnicos
iADX	Average Directional Index	Indicadores Técnicos

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
iADXWilder	Average Directional Index by Welles Wilder	Indicadores Técnicos
iAlligator	Alligator	Indicadores Técnicos
iAMA	Adaptive Moving Average	Indicadores Técnicos
iAO	Awesome Oscillator	Indicadores Técnicos
iATR	Average True Range	Indicadores Técnicos
iBands	Bollinger Bands®	Indicadores Técnicos
iBearsPower	Bears Power	Indicadores Técnicos
iBullsPower	Bulls Power	Indicadores Técnicos
iBWMFI	Market Facilitation Index by Bill Williams	Indicadores Técnicos
iCCI	Commodity Channel Index	Indicadores Técnicos
iChaikin	Chaikin Oscillator	Indicadores Técnicos
iCustom	Custom indicator	Indicadores Técnicos
iDEMA	Double Exponential Moving Average	Indicadores Técnicos
iDeMarker	DeMarker	Indicadores Técnicos
iEnvelopes	Envelopes	Indicadores Técnicos
iForce	Force Index	Indicadores Técnicos
iFractals	Fractals	Indicadores Técnicos
iFrAMA	Fractal Adaptive Moving Average	Indicadores Técnicos
iGator	Gator Oscillator	Indicadores Técnicos
iIchimoku	Ichimoku Kinko Hyo	Indicadores Técnicos
iMA	Moving Average	Indicadores Técnicos
iMACD	Moving Averages Convergence-Divergence	Indicadores Técnicos
iMFI	Money Flow Index	Indicadores Técnicos
iMomentum	Momentum	Indicadores Técnicos
IndicatorCreate	Retorna o handle para o indicador técnico especificado criado por um array de parâmetros de tipo MqlParam	Séries Temporais e Acesso a Indicadores

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
IndicatorParameters	Baseado no handle especificado, retorna o número de parâmetros de entrada do indicador, bem como os valores e tipos dos parâmetros	Séries Temporais e Acesso a Indicadores
IndicatorRelease	Remove um handle de indicador e libera o bloco de cálculo do indicador, se ele não for usado por ninguém mais	Séries Temporais e Acesso a Indicadores
IndicatorSetDouble	Define o valor da propriedade de um indicador do tipo double	Indicadores Customizados
IndicatorSetInteger	Define o valor da propriedade de um indicador do tipo int	Indicadores Customizados
IndicatorSetString	Define o valor da propriedade de um indicador do tipo string	Indicadores Customizados
IntegerToString	Converte int em uma string de comprimento predefinido	Funções de Conversão
iOBV	On Balance Volume	Indicadores Técnicos
iOsMA	Moving Average of Oscillator (MACD histogram)	Indicadores Técnicos
iRSI	Relative Strength Index	Indicadores Técnicos
iRVI	Relative Vigor Index	Indicadores Técnicos
iSAR	Parabolic Stop And Reverse System	Indicadores Técnicos
IsStopped	Retorna true, se um programa mql5 foi ordenado a interromper sua operação	Verificando Estado
iStdDev	Standard Deviation	Indicadores Técnicos
iStochastic	Stochastic Oscillator	Indicadores Técnicos
iTEMA	Triple Exponential Moving Average	Indicadores Técnicos
iTriX	Triple Exponential Moving Averages Oscillator	Indicadores Técnicos
iVIDyA	Variable Index Dynamic Average	Indicadores Técnicos
iVolumes	Volumes	Indicadores Técnicos
iWPR	Williams' Percent Range	Indicadores Técnicos
log	Retorna o logaritmo natural	Funções Matemáticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
log10	Retorna o logaritmo decimal (de um número na base 10)	Funções Matemáticas
MarketBookAdd	Fornece abertura da Profundidade de Mercado para um ativo selecionado, e subscrever para receber notificações de alterações do DOM (Depth of Market)	Informações de Mercado
MarketBookGet	Retorna um array de estrutura MqlBookInfo contendo registros da Profundidade de Mercado de uma ativo especificado	Informações de Mercado
MarketBookRelease	Fornece fechamento da Profundidade de Mercado para um ativo selecionado, e cancelar a subscrição para receber notificações de alterações do DOM	Informações de Mercado
MathAbs	Retorna o valor absoluto (modulo) do valor numérico especificado	Funções Matemáticas
MathArccos	Retorna o arco-cosseno de x em radianos	Funções Matemáticas
MathArcsin	Retorna o arco-seno de x em radianos	Funções Matemáticas
MathArctan	Retorna o arco-tangente de x em radianos	Funções Matemáticas
MathCeil	Retorna o valor numérico inteiro mais próximo do acima	Funções Matemáticas
MathCos	Retorna o cosseno de um número	Funções Matemáticas
MathExp	Retorna o exponencial de um número	Funções Matemáticas
MathFloor	Retorna o valor numérico inteiro mais próximo do abaixo	Funções Matemáticas
MathIsValidNumber	Verifica a validade de um número real	Funções Matemáticas
MathLog	Retorna o logaritmo natural	Funções Matemáticas
MathLog10	Retorna o logaritmo decimal (de um número na base 10)	Funções Matemáticas
MathMax	Retorna o valor máximo entre dois valores numéricos	Funções Matemáticas

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
MathMin	Retorna de valor mínimo entre dois valores numéricos	Funções Matemáticas
MathMod	Retorna o resto real da divisão de dois números	Funções Matemáticas
MathPow	Eleva uma base a um expoente especificado	Funções Matemáticas
MathRand	Retorna um valor pseudo-aleatório entre 0 e 32767	Funções Matemáticas
MathRound	Arredonda um valor para o inteiro mais próximo	Funções Matemáticas
MathSin	Retorna o seno de um número	Funções Matemáticas
MathSqrt	Retorna a raiz quadrada	Funções Matemáticas
MathSrand	Define o ponto inicial para geração de uma série de inteiros pseudo-aleatórios	Funções Matemáticas
MathTan	Retorna a tangente de um número	Funções Matemáticas
MessageBox	Cria, exibe uma caixa de mensagem e o gerencia	Funções Comuns
MQLInfoInteger	Retorna um valor inteiro de uma propriedade correspondente de um programa mql5 em execução	Verificando Estado
MQLInfoString	Retorna um valor de string de uma propriedade correspondente de um programa mql5 em execução	Verificando Estado
MT5Initialize	Estabelece a conexão com o terminal MetaTrader 5	MetaTrader para Python
MT5Shutdown	Fecha a conexão estabelecida anteriormente com o terminal MetaTrader 5	MetaTrader para Python
MT5TerminalInfo	Obtém o estado e os parâmetros do terminal MetaTrader 5 conectado	MetaTrader para Python
MT5Version	Retorna a versão do terminal MetaTrader 5	MetaTrader para Python
MT5CopyRatesFrom	Recebe barras do terminal MetaTrader 5, a partir da data especificada	MetaTrader para Python

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
MT5CopyRatesFromPos	Recebe barras do terminal MetaTrader 5, a partir do índice especificado	MetaTrader para Python
MT5CopyRatesRange	Recebe barras a partir do terminal MetaTrader 5, no intervalo de datas especificado	MetaTrader para Python
MT5CopyTicksFrom	Recebe ticks do terminal MetaTrader 5, a partir da data especificada	MetaTrader para Python
MT5CopyTicksRange	Recebe ticks a partir do terminal MetaTrader 5, no intervalo de datas especificado	MetaTrader para Python
NormalizeDouble	Arredonda um número de ponto flutuando para uma precisão especificada	Funções de Conversão
ObjectCreate	Cria um objecto de tipo específico para um gráfico específico	Funções de Objeto
ObjectDelete	Remove o objeto com o nome específico a partir de um gráfico específico (a partir de gráfico de sub-janela específica)	Funções de Objeto
ObjectFind	Pesquisa um objeto com o ID especificado pelo nome	Funções de Objeto
ObjectGetDouble	Retorna o valor double da propriedade do objeto correspondente	Funções de Objeto
ObjectGetInteger	Retorna o valor integer da propriedade do objeto correspondente	Funções de Objeto
ObjectGetString	Retorna o valor string da propriedade do objeto correspondente	Funções de Objeto
ObjectGetTimeByValue	Retorna o valor de tempo ao valor do preço do objeto especifico	Funções de Objeto
ObjectGetValueByTime	Retorna o valor do preço de um objeto ao período especificado	Funções de Objeto
ObjectMove	Muda as coordenadas do ponto de ancoragem do objeto especificado	Funções de Objeto
ObjectName	Retorna o nome de um objeto do tipo correspondente para um	Funções de Objeto

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	gráfico específico (gráfico de sub-janela específica)	
ObjectsDeleteAll	Remove todos os objetos com o tipo específico a partir de um gráfico específico (a partir de gráfico de sub-janela específica)	Funções de Objeto
ObjectSetDouble	Define o valor da propriedade do objeto correspondente	Funções de Objeto
ObjectSetInteger	Define o valor da propriedade do objeto correspondente	Funções de Objeto
ObjectSetString	Define o valor da propriedade do objeto correspondente	Funções de Objeto
ObjectsTotal	Retorna o número de objetos do tipo especifico no gráfico especificado (gráfico de sub-janela específica)	Funções de Objeto
OnStart	É chamada no script quando ocorre um evento Start para executar ações definidas no script	Manipulação de eventos
OnInit	É chamada em indicadores e EAs quando ocorre um evento Init para inicializar um programa MQL5 em execução	Manipulação de eventos
OnDeinit	É chamada em indicadores e EAs quando ocorre um evento Deinit para desinicializar um programa MQL5 em execução	Manipulação de eventos
OnTick	É chamada em EAs quando ocorre um evento NewTick para processar uma nova cotação	Manipulação de eventos
OnCalculate	É chamada em indicadores quando ocorre um evento Calculate para processar alterações de dados de preço	Manipulação de eventos
OnTimer	É chamada em indicadores e EAs no início de um evento periódico Timer que é gerado pelo terminal com o intervalo de tempo definido	Manipulação de eventos
OnTrade	É chamada em EAs quando ocorre um evento Trade que é gerado no	Manipulação de eventos

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	final da negociação no servidor de negociação	
OnTradeTransaction	É chamada em EAs quando ocorre um evento TradeTransaction para processar os resultados de uma solicitação de negociação	Manipulação de eventos
OnBookEvent	É chamada em EAs quando ocorre um evento BookEvent para processar alterações no livro de ofertas	Manipulação de eventos
OnChartEvent	É chamada em indicadores e Eas quando ocorre um evento ChartEvent para processar mudanças no gráfico causadas por ações do usuário ou pelo funcionamento de programas MQL5	Manipulação de eventos
OnTester	É chamada em EAs quando ocorre um evento Tester para executar as ações necessárias no final do teste	Manipulação de eventos
OnTesterInit	É chamada em EAs quando ocorre um evento TesterInit para executar as ações necessárias antes do início da otimização	Manipulação de eventos
OnTesterDeinit	É chamada em EAs quando ocorre um evento TesterDeinit para realizar as ações necessárias após a otimização do EA	Manipulação de eventos
OnTesterPass	É chamada em EAs quando ocorre um evento TesterPass para processar a chegada de um novo quadro de dados durante a otimização do EA	Manipulação de eventos
OrderCalcMargin	Calcula a margem necessária para o tipo de ordem especificado, na moeda de depósito	Funções de Negociação (Trade)
OrderCalcProfit	Calcula o lucro com base nos parâmetros passados, na moeda de depósito	Funções de Negociação (Trade)
OrderCheck	Verifica se há fundos suficientes para executar a operação de	Funções de Negociação (Trade)

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	negócio necessária.	
OrderGetDouble	Retorna a propriedade solicitada da ordem (double)	Funções de Negociação (Trade)
OrderGetInteger	Retorna a propriedade solicitada da ordem (datetime ou int)	Funções de Negociação (Trade)
OrderGetString	Retorna a propriedade solicitada da ordem (string)	Funções de Negociação (Trade)
OrderGetTicket	Devolver o bilhete de uma ordem correspondente	Funções de Negociação (Trade)
OrderSelect	Seleciona uma ordem para posteriormente trabalhar com ele	Funções de Negociação (Trade)
OrderSend	Envia pedidos de negócios para um servidor	Funções de Negociação (Trade)
OrderSendAsync	De forma assíncrona envia pedidos de comércio sem esperar a resposta da negociação do servidor de negócio	Funções de Negociação (Trade)
OrdersTotal	Retorna o número de ordens	Funções de Negociação (Trade)
ParameterGetRange	Recebe os dados na faixa de valores e o passo de alteração para uma variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia	Trabalhando com Resultados de Otimização
ParameterSetRange	Especifica o uso de variável de entrada ao otimizar um Expert Advisor no Testador de Estratégia: valor, mudança de passo, os valores iniciais e finais	Trabalhando com Resultados de Otimização
Period	Retorna a janela de tempo do gráfico corrente	Verificando Estado
PeriodSeconds	Retorna o número de segundos no período	Funções Comuns
PlaySound	Toca um arquivo de som	Funções Comuns
PlotIndexGetInteger	Retorna o valor da propriedade de linha de um indicador do tipo inteiro	Indicadores Customizados
PlotIndexSetDouble	Define o valor da propriedade de linha de um indicador do tipo	Indicadores Customizados

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	double	
PlotIndexSetInteger	Define o valor da propriedade de linha de um indicador do tipo int	Indicadores Customizados
PlotIndexSetString	Define o valor da propriedade de linha de um indicador do tipo string	Indicadores Customizados
Point	Retorna o tamanho do ponto do símbolo atual na moeda de cotação.	Verificando Estado
PositionGetDouble	Retorna a propriedade solicitado de uma posição aberta (double)	Funções de Negociação (Trade)
PositionGetInteger	Retorna a propriedade solicitada uma posição aberta (datetime ou int)	Funções de Negociação (Trade)
PositionGetString	Retorna a propriedade solicitada uma posição aberta (string)	Funções de Negociação (Trade)
PositionGetSymbol	Retorna o símbolo correspondente à posição aberta	Funções de Negociação (Trade)
PositionSelect	Escolhe uma posição aberta para posteriormente trabalhar com ele	Funções de Negociação (Trade)
PositionsTotal	Retorna o número de posições em aberto	Funções de Negociação (Trade)
pow	Eleva uma base a um expoente especificado	Funções Matemáticas
Print	Exibe uma mensagem no log	Funções Comuns
PrintFormat	Formata e imprime um conjunto de símbolos e valores em um arquivo de log de acordo com um formato predefinido	Funções Comuns
rand	Retorna um valor pseudo-aleatório entre 0 e 32767	Funções Matemáticas
ResetLastError	Define o valor da variável predeterminada _LastError para zero	Funções Comuns
ResourceCreate	Cria um recurso de imagem baseado em um conjunto de dados	Funções Comuns
ResourceFree	Exclui recursos criados dinamicamente (liberando a	Funções Comuns

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	memória alocada para ele)	
ResourceReadImage	Lê os dados do recurso gráfico criado pela função ResourceCreate() ou salvo em arquivo EX5 durante a compilação	Funções Comuns
ResourceSave	Salva um recurso em um arquivo especificado	Funções Comuns
round	Arredonda um valor para o inteiro mais próximo	Funções Matemáticas
SendFTP	Envia um arquivo no endereço especificado na janela de configurações da guia "FTP"	Funções Comuns
SendMail	Envia um email no endereço especificado na janela de configurações da guia "Email"	Funções Comuns
SendNotification	Envia notificações push para terminais móveis, cujos MetaQuotes ID's estão especificados na guia "Notificações"	Funções Comuns
SeriesInfoInteger	Retorna informação sobre o estado dos dados históricos	Séries Temporais e Acesso a Indicadores
SetIndexBuffer	Vincula-se o buffer de indicador especificado com dinâmica unidimensional array do tipo double	Indicadores Customizados
ShortArrayToString	Copia parte de um array em uma string	Funções de Conversão
ShortToString	Converte código de símbolo (unicode) em string de um símbolo	Funções de Conversão
SignalBaseGetDouble	Returns the value of double type property for selected signal	Trade Signals
SignalBaseGetInteger	Returns the value of integer type property for selected signal	Trade Signals
SignalBaseGetString	Returns the value of string type property for selected signal	Trade Signals
SignalBaseSelect	Selects a signal from signals, available in terminal for further working with it	Trade Signals

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
SignalBaseTotal	Returns the total amount of signals, available in terminal	Trade Signals
SignalInfoGetDouble	Returns the value of double type property of signal copy settings	Trade Signals
SignalInfoGetInteger	Returns the value of integer type property of signal copy settings	Trade Signals
SignalInfoGetString	Returns the value of string type property of signal copy settings	Trade Signals
SignalInfoSetDouble	Sets the value of double type property of signal copy settings	Trade Signals
SignalInfoSetInteger	Sets the value of integer type property of signal copy settings	Trade Signals
SignalSubscribe	Subscribes to the trading signal	Trade Signals
SignalUnsubscribe	Cancels subscription	Trade Signals
sin	Retorna o seno de um número	Funções Matemáticas
Sleep	Suspende por um intervalo especificado a execução do Expert Advisor corrente ou script	Funções Comuns
SocketCreate	Cria um soquete com os sinalizadores especificados e retorna seu identificador	Funções de rede
SocketClose	Fecha o soquete	Funções de rede
SocketConnect	Conecta-se ao servidor com controle de tempo limite	Funções de rede
SocketIsConnected	Verifica se há um soquete conectado atualmente	Funções de rede
SocketIsReadable	Obtém o número de bytes que podem ser lidos do soquete.	Funções de rede
SocketIsWritable	Verifica se os dados podem ser registrados no soquete no momento atual.	Funções de rede
SocketTimeouts	Define os tempos limite a fim de receber e enviar dados para o objeto de sistema de soquete	Funções de rede
SocketRead	Lê dados do soquete	Funções de rede
SocketSend	Registra dados no soquete	Funções de rede

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
SocketTlsHandshake	Inicia uma conexão TLS (SSL) segura com o host especificado usando o protocolo TLS Handshake	Funções de rede
SocketTlsCertificate	Recebe informações sobre o certificado usado para proteger a conexão de rede.	Funções de rede
SocketTlsRead	Lê dados de uma conexão TLS segura	Funções de rede
SocketTlsReadAvailable	Lê todos os dados disponíveis a partir de uma conexão TLS segura	Funções de rede
SocketTlsSend	Envia dados por meio de uma conexão TLS segura	Funções de rede
sqrt	Retorna a raiz quadrada	Funções Matemáticas
srand	Define o ponto inicial para geração de uma série de inteiros pseudo-aleatórios	Funções Matemáticas
StringAdd	Adiciona uma string no final de uma outra string	Funções de String
StringBufferLen	Retorna o tamanho de buffer alocado para uma string	Funções de String
StringCompare	Compara duas string e retorna 1 se a primeira string for maior que a segunda; 0 - se as string são iguais; -1 (menos 1) - se a primeira string for menor que a segunda.	Funções de String
StringConcatenate	Forma uma string dos parâmetros passados	Funções de String
StringFill	Preenche uma string especificada com símbolos selecionados	Funções de String
StringFind	Busca por uma substring em uma string	Funções de String
StringFormat	Converte um número em string de acordo com um formato predefinido	Funções de Conversão
StringGetCharacter	Retorna o valor de um número localizado na posição de string especificada	Funções de String

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
StringInit	Inicializa uma string com símbolos especificados e fornece o comprimento da string especificada	Funções de String
StringLen	Retorna o número de símbolos em uma string	Funções de String
StringReplace	Substitui todas as substrings encontradas em uma string por uma seqüência de símbolos	Funções de String
StringSetCharacter	Retorna uma cópia de uma string com o valor alterado de um símbolo na posição especificada	Funções de String
StringSplit	Obtém substrings a partir de um separador especificado da string especificada, retorna o número de substrings obtidas	Funções de String
StringSubstr	Extrai uma substring a partir de uma string de texto começando a partir de uma posição especificada	Funções de String
StringToCharArray	Copia símbolo a símbolo uma string convertida de Unicode para ANSI, para um lugar selecionado de um array de tipo uchar	Funções de Conversão
StringToColor	Converte uma string "R,G,B" ou uma string com nome de cor em valor de tipo color	Funções de Conversão
StringToDouble	Converte uma string contendo um representação de símbolo de número em um número de tipo double	Funções de Conversão
StringToInteger	Converte uma string contendo uma representação de símbolo de número em um número de tipo int	Funções de Conversão
StringToLower	Transforma todos os símbolos de uma string selecionada para minúscula pela localização	Funções de String
StringToShortArray	Copia símbolo a símbolo uma string para uma parte selecionada de array de tipo ushort	Funções de Conversão

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
StringToTime	Converte uma string contendo hora ou data no formato "yyyy.mm.dd [hh:mi]" em um tipo datetime	Funções de Conversão
StringToUpper	Transforma todos os símbolos de uma string selecionada para maiúscula pela localização	Funções de String
StringTrimLeft	Remove caracteres de quebra de linha, espaços e tabs na parte esquerda da string	Funções de String
StringTrimRight	Remove caracteres de quebra de linha, espaços e tabs na parte direita da string	Funções de String
StructToTime	Converte uma variável do tipo estrutura MqlDateTime em um valor datetime	Data e Hora
Symbol	Retorna o nome do ativo do gráfico corrente	Verificando Estado
SymbolInfoDouble	Retorna um valor double de um ativo para a propriedade correspondente	Informações de Mercado
SymbolInfoInteger	Retorna um valor de tipo inteiro (long, datetime, int ou bool) de um ativo especificado para a propriedade correspondente	Informações de Mercado
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction	Informações de Mercado
SymbolInfoSessionQuote	Permite obter a hora de início e fim das sessões de cotação especificadas para um ativo especificado e dia da semana.	Informações de Mercado
SymbolInfoSessionTrade	Permite obter a hora de início e fim das sessões de negociação especificadas para um ativo especificado e dia da semana.	Informações de Mercado
SymbolInfoString	Retorna um valor do tipo string de um ativo especificado para a propriedade correspondente	Informações de Mercado
SymbolInfoTick	Retorna os preços correntes para um ativo especificado em uma	Informações de Mercado

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
	variável do tipo MqlTick	
SymbolIsSynchronized	Verifica se dados de um ativo selecionado no terminal estão sincronizados com dados no servidor de negociações	Informações de Mercado
SymbolName	Retorna o nome de um ativo especificado	Informações de Mercado
SymbolSelect	Seleciona um ativo na janela Observação de Mercado ou remove um ativo desta janela	Informações de Mercado
SymbolsTotal	Retorna o número de ativos disponíveis (selecionados na janela Observação de Mercado ou todos)	Informações de Mercado
tan	Retorna a tangente de um número	Funções Matemáticas
TerminalClose	Ordena o terminal a finalizar a operação	Funções Comuns
TerminalInfoDouble	Retorna um valor de double de uma propriedade correspondente do ambiente de um programa mql5	Verificando Estado
TerminalInfoInteger	Retorna um valor inteiro de uma propriedade correspondente do ambiente de um programa mql5	Verificando Estado
TerminalInfoString	Retorna um valor de string de uma propriedade correspondente do ambiente de um programa mql5	Verificando Estado
TesterStatistics	Retorna o valor de uma estatística específica calculada com base em resultados de teste	Funções Comuns
TextGetSize	Retorna a largura e a altura da string nas configurações das fontes usadas no momento	Funções de Objeto
TextOut	Transfere o texto para um array personalizado (buffer) projetado para a criação de recursos gráficos	Funções de Objeto
TextSetFont	Define a fonte para exibir o texto usando métodos de desenho (Arial 20, usado por padrão)	Funções de Objeto

© 2000-2024, MetaQuotes Ltd.

Função	Ação	Section
TimeCurrent	Retorna a última hora conhecida do servidor (hora da última cotação recebida) no formato de datetime(	Data e Hora
TimeDaylightSavings	Retorna o sinal do ajuste do horário de verão	Data e Hora
TimeGMT	Retorna a hora GMT em formato datetime com o horário de verão por hora local do computador, de onde o terminal cliente está executando	Data e Hora
TimeGMTOffset	Retorna a diferença corrente entre a hora GMT e a hora do computador local em segundos, levando em consideração o ajuste do horário de verão	Data e Hora
TimeLocal	Retorna a hora do computador local em formato datetime	Data e Hora
TimeToString	Converte um valor contendo hora em segundos decorridos deste 01.01.1970 em uma string de formato "yyyy.mm.dd hh:mi"	Funções de Conversão
TimeToStruct	Converte um valor datetime em uma variável do tipo estrutura MqlDateTime	Data e Hora
TimeTradeServer	Retorna a hora calculada corrente do servidor de negociação	Data e Hora
UninitializeReason	Retorna o código do motivo da desinicialização	Verificando Estado
WebRequest	Sends HTTP request to the specified server	Funções Comuns
ZeroMemory	Redefine uma variável passada para ela por referência. A variável pode ser de qualquer tipo, exceto classes e estruturas que tenham construtores.	Funções Comuns

© 2000-2024, MetaQuotes Ltd.

Lista de Constantes MQL5

Todas as constantes MQL5 em ordem alfabética.

Constant	Description	Usage
__DATE__	Data de compilação do arquivo sem a hora (horas, minutos e segundos são iguais a 0)	Print
__DATETIME__	Data e hora de compilação do arquivo	Print
__FILE__	Nome do arquivo compilado corrente	Print
__FUNCSIG__	Assinatura da função em cujo corpo a macro está localizada. O registro da descrição completa de funções pode ser útil na identific	Print

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ação de funções sobrecarregadas	
__FUNCTION__	Nome da função, em cujo corpo a macro está localizada	Print
__LINE__	Assinatura da função em cujo corpo a macro está localizada.	Print
__MQLBUILD__, __MQL5BUILD__	Número do compilação	Print
__PATH__	O caminho absoluto para o arquivo que está sendo compilado correntemente	Print
ACCOUNT_ASSETS	The current assets of an account	AccountInfoDouble
ACCOUNT_BALANCE	Saldo da conta na moeda	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de depósito	
ACCOUNT_COMMISSION_BLOCKED	The current blocked commission amount on an account	AccountInfoDouble
ACCOUNT_COMPANY	Nome de uma empresa que serve a conta	AccountInfoString
ACCOUNT_CREDIT	Crédito da conta na moeda de depósito	AccountInfoDouble
ACCOUNT_CURRENCY	Moeda da conta	AccountInfoString
ACCOUNT_EQUITY	Saldo a mercado da conta na moeda de depósito	AccountInfoDouble
ACCOUNT_LEVERAGE	Alavancagem da conta	AccountInfoInteger
ACCOUNT_LIABILITIES	The current liabilities on an account	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	Número permitido máximo	AccountInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de ordens pendentes ativas	
ACCOUNT_LOGIN	Número da conta	AccountInfoInteger
ACCOUNT_MARGIN	Margem da conta usada na moeda de depósito	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Margem livre de uma conta na moeda de depósito	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	Initial margin. The amount reserved on an account to cover the margin of all pending orders	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Nível de margem de conta em percentagem	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	Maintenance margin. The minimum	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	equity reserved on an account to cover the minimum amount of all open positions	
ACCOUNT_MARGIN_SO_CALL	Nível de chamada de margem. Dependendo da definição, ACCOUNT_MARGIN_SO_MODE é expresso em percentagem ou em moeda de depósito	AccountInfoDouble
ACCOUNT_MARGIN_SO_MODE	Modo para definir a margem permitida mínima	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Nível de margem de Stop Out (encerramento forçado)	AccountInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	. Dependendo da definição, ACCOUNT_MARGIN_SO_MODE é expresso em percentagem ou em moeda de depósito	
ACCOUNT_NAME	Nome do cliente	AccountInfoString
ACCOUNT_PROFIT	Lucro atual de uma conta na moeda de depósito	AccountInfoDouble
ACCOUNT_SERVER	Nome do servidor de negociação	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Modo stop out da conta em dinheiro	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Modo stop out da conta em percentagem	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Negociação	AccountInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	permitida para a conta atual	
ACCOUNT_TRADE_EXPERT	Negociação permitida para um Expert Advisor	AccountInfoInteger
ACCOUNT_TRADE_MODE	Modo de negociação da conta	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Conta de torneio	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Conta de Demonstração	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Conta real	AccountInfoInteger
ALIGN_CENTER	Centrado (somente para o objeto Edit)	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	Alinhamento a esquerda	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	Alinhamento a direita	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Ponto de ancoragem rigorosamente no centro	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	do objeto	
ANCHOR_LEFT	Ponto de ancoragem para a esquerda no centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Ponto de ancoragem no canto inferior esquerdo	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Ponto de ancoragem no canto superior esquerdo	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Ponto de ancoragem abaixo no centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Ponto de ancoragem para a direita no centro	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	Ponto de ancoragem no canto inferior direito	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Ponto de ancorag	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	em no canto superior direito	
ANCHOR_UPPER	Ponto de ancoragem acima no centro	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Linha principal	Linhas de Indicadores
BOOK_TYPE_BUY	Ordem de compra (Bid)	MqlBookInfo
BOOK_TYPE_BUY_MARKET	Ordem de venda (Offer)	MqlBookInfo
BOOK_TYPE_SELL	Ordem de venda (Offer)	MqlBookInfo
BOOK_TYPE_SELL_MARKET	Ordem de venda (Offer)	MqlBookInfo
BORDER_FLAT	Forma lisa	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	Forma em relevo	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Forma cavada	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	Valor máximo, que pode ser representado	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	pelo tipo char	
CHAR_MIN	Valor mínimo, que pode ser representado pelo tipo char	Constantes de Tipo Numérico
CHART_AUTOSCROLL	Modo de movimentação automática para a borda direita do gráfico	ChartSetInteger, ChartGetInteger
CHART_BARS	Exibe como uma seqüência de barras	ChartSetInteger
CHART_BEGIN	Começo do gráfico (o preço mais antigo)	ChartNavigate
CHART_BRING_TO_TOP	Exibe o gráfico no topo de outros gráficos	ChartSetInteger, ChartGetInteger
CHART_CANDLES	Exibe como candles japoneses	ChartSetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
CHART_COLOR_ASK	Cor de nível de preço de compra	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	Cor de fundo do gráfico	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	Cor de nível de preço de venda	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	Cor de corpo de um candle de baixa	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	Cor de corpo de um candle de alta	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	Cor para a barra de baixa, sombras e bordas de candles de baixa	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	Cor de gráfico de linha e cor de candles "Doji" japoneses	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	Cor para a barra de alta, sombras e bordas	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de candles de alta	
CHART_COLOR_FOREGROUND	Cor dos eixos, escalas e linha OHLC	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	Cor da grade	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	Cor de linha do último preço de operação executada	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	Cor de níveis de ordem de stop (Stop Loss e Take Profit)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	Cor dos volumes e níveis de abertura de posição	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Texto de um comentário em um gráfico	ChartSetString, ChartGetString
CHART_CURRENT_POS	Posição corrente	ChartNavigate

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
CHART_DRAG_TRADE_LEVELS	Permissão para arrastar os níveis de negociação num gráfico com um mouse. O modo de arrastar é ativado pelo padrão (valor verdadeiro)	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	Eventos CHARTEVENT_MOUSE_MOVE) envia notificações de movimento de mouse e clique de mouse a todos os programas MQL5 num gráfico	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_CREATE	Envia uma notificação de um evento de criação	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de novo objeto (CHART_EVENT_OBJECT_CREATE) para todos os programas MQL5 em um gráfico	
CHART_EVENT_OBJECT_DELETE	Envia uma notificação de um evento de exclusão de objeto (CHART_EVENT_OBJECT_DELETE) para todos os programas MQL5 em um gráfico	ChartSetInteger, ChartGetInteger
CHART_FIRST_VISIBLE_BAR	Número das primeiras barras visíveis no gráfico. A indexação de barras é a mesma para	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	séries de tempo.	
CHART_FIXED_MAX	Máximo fixo do gráfico	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Mínimo fixo do gráfico	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	Posição fixa do gráfico a partir da borda esquerda em valor percentual. A posição fixa do gráfico é marcada por um pequeno triângulo cinza no eixo do tempo horizontal. Ele é exibido somente se a rolagem automática do gráfico para a direita na chegada de um tick (preço novo)	ChartSetDouble, ChartGetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	estiver desabilitada (veja a propriedade CHART_AUTOSCROLL). A barra em uma posição fixa permanece no mesmo lugar ao se ampliar e reduzir o gráfico.	
CHART_FOREGROUND	Gráfico de preços no primeiro plano	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	Altura de gráfico em pixels	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	Identificado objeto de "Gráfico" (OBJ_CHART) - retorna verdadeiro para um objeto gráfico.	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	Retorna falso para um gráfico real	
CHART_LINE	Exibe como uma linha desenha por preços de fechamento	ChartSetInteger
CHART_MODE	Tipo de gráfico (candles, barras ou linha)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Rolagem do gráfico horizontalmente usando o botão esquerdo do mouse. Rolagem vertical também está disponível se for definido como verdadeiro o valor de todas as seguintes propriedades:	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	CHART_SCALEFIX, CHART_SCALEFIX_11 ou CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	Escala em pontos por barra	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Máximo do gráfico	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Mínimo do gráfico	ChartSetDouble, ChartGetDouble
CHART_SCALE	Escala	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	Escala para ser especificada em pontos por barra	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Modo de escala fixo	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	Modo de escala 1:1	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Modo de encaixe do gráfico de preço a partir da borda direita	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
CHART_SHIFT_SIZE	O tamanho da barra zero encaixada a partir da borda direita em porcentagens	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	Exibe valores de preço de compra como uma linha horizontal em um gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_BID_LINE	Exibe valores de preço de venda como uma linha horizontal em um gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	Exibição da escala de tempo em um gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	Exibe grade no	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	gráfico	
CHART_SHOW_LAST_LINE	Exibe valores de último preço como uma linha horizontal em um gráfico	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Descrições pop-up de objetos gráficos	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	Exibe valores OHLC no canto superior esquerdo	ChartSetInteger, ChartGetInteger
CHART_SHOW_ONE_CLICK	Showing the "One click trading" panel on a chart	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	Exibe separadores verticais entre períodos adjacentes	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	Exibição da escala de preço	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	em um gráfico	
CHART_SHOW_TRADE_LEVELS	Exibição de níveis de negociação no gráfico (níveis de abertura de posição, Stop Loss, Take Profit e ordens pendentes	ChartSetInteger, ChartGetInteger
CHART_SHOW_VOLUMES	Exibe volume no gráfico	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	O número de barras no gráfico que podem ser exibidas	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	Volumes não são exibidos	ChartSetInteger
CHART_VOLUME_REAL	Volume de negociação	ChartSetInteger
CHART_VOLUME_TICK	Volumes de tick	ChartSetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	(mudança de preço)	
CHART_WIDTH_IN_BARS	Largura de gráfico em barras	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	Largura de gráfico em pixels	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Manuseio da janela do gráfico (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	Visibilidade de sub-janelas	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	A distância entre a moldura superior da sub-janela do indicador e a moldura superior da janela do gráfico principal, ao longo do eixo Y vertical,	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	in pixels. No caso de um evento de mouse, as coordenadas do cursor são passadas em termos das coordenadas da janela do gráfico principal, enquanto que as coordenadas de objetos gráficos em uma sub-janela do indicador são definidas em relação ao canto superior esquerdo da sub-janela. O valor é requerido para	

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	converter as coordenadas absolutas do gráfico principal em coordenadas locais da sub-janela para correto trabalho com os objetos gráficos, cujas coordenadas são definidas em relação ao canto superior esquerdo da moldura da sub-janela.	
CHART_WINDOWS_TOTAL	O número total de janelas de gráficos, incluindo sub-janelas de indicadores	ChartSetInteger, ChartGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
CHARTEVENT_CHART_CHANGE	Alteração do tamanho do gráfico ou modificação das propriedades do gráfico através da janela de Propriedades	OnChartEvent
CHARTEVENT_CLICK	Dar um clique em um gráfico	OnChartEvent
CHARTEVENT_CUSTOM	O número inicial de um evento de uma faixa de eventos customizados	OnChartEvent
CHARTEVENT_CUSTOM_LAST	O número final de um evento de uma faixa de eventos customizados	OnChartEvent
CHARTEVENT_KEYDOWN	Teclada	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Movimento de mouse,	OnChartEvent

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	cliques de mouse (se CHART_EVENT_MOUSE_MOVE=true é definido para o gráfico)	
CHARTEVENT_OBJECT_CHANGE	Propriedade do objeto gráfico alterada via janela de diálogo de propriedades	OnChartEvent
CHARTEVENT_OBJECT_CLICK	Dar um clique em um objeto gráfico	OnChartEvent
CHARTEVENT_OBJECT_CREATE	Objeto gráfico criado (se CHART_EVENT_OBJECT_CREATE =true é definido para o gráfico)	OnChartEvent
CHARTEVENT_OBJECT_DELETE	Objeto gráficoexcluído (se	OnChartEvent

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	CHART_EVENT_OBJECT_DELETE= true é definido para o gráfico)	
CHARTEVENT_OBJECT_DRAG	Arrastar e soltar um objeto gráfico	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	Fim de edição de texto no objeto gráfico Edit	OnChartEvent
CHARTS_MAX	O número possível máximo de gráficos abertos simultaneamente em um terminal	Outras Constantes
CHIKOUSPAN_LINE	Linha Chikou Span	Linhas de Indicadores
clrAliceBlue	Alice Blue	Cores Web
clrAntiqueWhite	Antique White	Cores Web
clrAqua	Aqua	Cores Web
clrAquamarine	Aquamarine	Cores Web
clrBeige	Beige	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrBisque	Bisque	Cores Web
clrBlack	Black	Cores Web
clrBlanchedAlmond	Blanched Almond	Cores Web
clrBlue	Blue	Cores Web
clrBlueViolet	Blue Violet	Cores Web
clrBrown	Brown	Cores Web
clrBurlyWood	Burly Wood	Cores Web
clrCadetBlue	Cadet Blue	Cores Web
clrChartreuse	Chartreuse	Cores Web
clrChocolate	Chocolate	Cores Web
clrCoral	Coral	Cores Web
clrCornflowerBlue	Cornflower Blue	Cores Web
clrCornsilk	Cornsilk	Cores Web
clrCrimson	Crimson	Cores Web
clrDarkBlue	Dark Blue	Cores Web
clrDarkGoldenrod	Dark Goldenrod	Cores Web
clrDarkGray	Dark Gray	Cores Web
clrDarkGreen	Dark Green	Cores Web
clrDarkKhaki	Dark Khaki	Cores Web
clrDarkOliveGreen	Dark Olive Green	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrDarkOrange	Dark Orange	Cores Web
clrDarkOrchid	Dark Orchid	Cores Web
clrDarkSalmon	Dark Salmon	Cores Web
clrDarkSeaGreen	Dark Sea Green	Cores Web
clrDarkSlateBlue	Dark Slate Blue	Cores Web
clrDarkSlateGray	Dark Slate Gray	Cores Web
clrDarkTurquoise	Dark Turquoise	Cores Web
clrDarkViolet	Dark Violet	Cores Web
clrDeepPink	Deep Pink	Cores Web
clrDeepSkyBlue	Deep Sky Blue	Cores Web
clrDimGray	Dim Gray	Cores Web
clrDodgerBlue	Dodger Blue	Cores Web
clrFireBrick	Fire Brick	Cores Web
clrForestGreen	Forest Green	Cores Web
clrGainsboro	Gainsboro	Cores Web
clrGold	Gold	Cores Web
clrGoldenrod	Goldenrod	Cores Web
clrGray	Gray	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrGreen	Green	Cores Web
clrGreenYellow	Green Yellow	Cores Web
clrHoneydew	Honeydew	Cores Web
clrHotPink	Hot Pink	Cores Web
clrIndianRed	Indian Red	Cores Web
clrIndigo	Indigo	Cores Web
clrIvory	Ivory	Cores Web
clrKhaki	Khaki	Cores Web
clrLavender	Lavender	Cores Web
clrLavenderBlush	Lavender Blush	Cores Web
clrLawnGreen	Lawn Green	Cores Web
clrLemonChiffon	Lemon Chiffon	Cores Web
clrLightBlue	Light Blue	Cores Web
clrLightCoral	Light Coral	Cores Web
clrLightCyan	Light Cyan	Cores Web
clrLightGoldenrod	Light Goldenrod	Cores Web
clrLightGray	Light Gray	Cores Web
clrLightGreen	Light Green	Cores Web
clrLightPink	Light Pink	Cores Web
clrLightSalmon	Light Salmon	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrLightSeaGreen	Light Sea Green	Cores Web
clrLightSkyBlue	Light Sky Blue	Cores Web
clrLightSlateGray	Light Slate Gray	Cores Web
clrLightSteelBlue	Light Steel Blue	Cores Web
clrLightYellow	Light Yellow	Cores Web
clrLime	Lime	Cores Web
clrLimeGreen	Lime Green	Cores Web
clrLinen	Linen	Cores Web
clrMagenta	Magenta	Cores Web
clrMaroon	Maroon	Cores Web
clrMediumAquamarine	Medium Aquamarine	Cores Web
clrMediumBlue	Medium Blue	Cores Web
clrMediumOrchid	Medium Orchid	Cores Web
clrMediumPurple	Medium Purple	Cores Web
clrMediumSeaGreen	Medium Sea Green	Cores Web
clrMediumSlateBlue	Medium Slate Blue	Cores Web
clrMediumSpringGreen	Medium Spring Green	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrMediumTurquoise	Medium Turquoise	Cores Web
clrMediumVioletRed	Medium Violet Red	Cores Web
clrMidnightBlue	Midnight Blue	Cores Web
clrMintCream	Mint Cream	Cores Web
clrMistyRose	Misty Rose	Cores Web
clrMoccasin	Moccasin	Cores Web
clrNavajoWhite	Navajo White	Cores Web
clrNavy	Navy	Cores Web
clrNONE	Ausência de cor	Outras Constantes
clrOldLace	Old Lace	Cores Web
clrOlive	Olive	Cores Web
clrOliveDrab	Olive Drab	Cores Web
clrOrange	Orange	Cores Web
clrOrangeRed	Orange Red	Cores Web
clrOrchid	Orchid	Cores Web
clrPaleGoldenrod	Pale Goldenrod	Cores Web
clrPaleGreen	Pale Green	Cores Web
clrPaleTurquoise	Pale Turquoise	Cores Web
clrPaleVioletRed	Pale Violet	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	Red	
clrPapayaWhip	Papaya Whip	Cores Web
clrPeachPuff	Peach Puff	Cores Web
clrPeru	Peru	Cores Web
clrPink	Pink	Cores Web
clrPlum	Plum	Cores Web
clrPowderBlue	Powder Blue	Cores Web
clrPurple	Purple	Cores Web
clrRed	Red	Cores Web
clrRosyBrown	Rosy Brown	Cores Web
clrRoyalBlue	Royal Blue	Cores Web
clrSaddleBrown	Saddle Brown	Cores Web
clrSalmon	Salmon	Cores Web
clrSandyBrown	Sandy Brown	Cores Web
clrSeaGreen	Sea Green	Cores Web
clrSeashell	Seashell	Cores Web
clrSienna	Sienna	Cores Web
clrSilver	Silver	Cores Web
clrSkyBlue	Sky Blue	Cores Web
clrSlateBlue	Slate Blue	Cores Web
clrSlateGray	Slate Gray	Cores Web
clrSnow	Snow	Cores Web
clrSpringGreen	Spring Green	Cores Web

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
clrSteelBlue	Steel Blue	Cores Web
clrTan	Tan	Cores Web
clrTeal	Teal	Cores Web
clrThistle	Thistle	Cores Web
clrTomato	Tomato	Cores Web
clrTurquoise	Turquoise	Cores Web
clrViolet	Violet	Cores Web
clrWheat	Wheat	Cores Web
clrWhite	White	Cores Web
clrWhiteSmoke	White Smoke	Cores Web
clrYellow	Yellow	Cores Web
clrYellowGreen	Yellow Green	Cores Web
CORNER_LEFT_LOWER	Centro de coordenadas está no canto inferior esquerdo do gráfico	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Centro de coordenadas está no canto superior esquerdo do gráfico	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	Centro de coorden	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	adas está no canto inferior direito do gráfico	
CORNER_RIGHT_UPPER	Centro de coordenadas está no canto superior direito do gráfico	ObjectSetInteger, ObjectGetInteger
CP_ACP	O código de página ANSI do Windows corrente.	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	O código de página do sistema Macintosh corrente. Note: Este valor é usado principalmente em códigos de programa criados antigam	CharArrayToString, StringToCharArray, FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ente e está sem uso atualmente, já que computadores Macintosh modernos usando Unicode para codificação.	
CP_OEMCP	O código de página do sistema OEM corrente.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	Código de página de símbolo	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	O código de página ANSI do Windows para o thread corrente.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	Código de página UTF-7.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	Código de	CharArrayToString, StringToCharArray, FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	página UTF-8.	
CRYPT_AES128	AES encryption with 128 bit key (16 bytes)	CryptEncode, CryptDecode
CRYPT_AES256	AES encryption with 256 bit key (32 bytes)	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	ZIP archives	CryptEncode, CryptDecode
CRYPT_BASE64	BASE64	CryptEncode, CryptDecode
CRYPT_DES	DES encryption with 56 bit key (7 bytes)	CryptEncode, CryptDecode
CRYPT_HASH_MD5	MD5 HASH caculation	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	SHA1 HASH caculation	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	SHA256 HASH caculation	CryptEncode, CryptDecode
DBL_DIG	Número de algarismos decimais significativos	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	para o tipo double	
DBL_EPSILON	Valor mínimo, que satisfaz a condição: 1.0+DBL_EPSILON != 1.0 (para o tipo double)	Constantes de Tipo Numérico
DBL_MANT_DIG	Quantidade de bits em uma mantissa para o tipo double	Constantes de Tipo Numérico
DBL_MAX	Valor máximo, que pode ser representado pelo tipo double	Constantes de Tipo Numérico
DBL_MAX_10_EXP	Valor decimal máximo de grau de expoente para o tipo double	Constantes de Tipo Numérico
DBL_MAX_EXP	Valor binário máximo de grau	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de expoente para o tipo double	
DBL_MIN	Valor positivo mínimo, que pode ser representado pelo tipo double	Constantes de Tipo Numérico
DBL_MIN_10_EXP	Valor decimal mínimo de grau de expoente para o tipo double	Constantes de Tipo Numérico
DBL_MIN_EXP	Valor binário mínimo de grau de expoente para o tipo double	Constantes de Tipo Numérico
DEAL_COMMENT	Comentário de uma operação	HistoryDealGetString
DEAL_COMMISSION	Comissão de uma operação	HistoryDealGetDouble
DEAL_ENTRY	Entrada de uma	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	operação - entrada, saída, reversão	
DEAL_ENTRY_IN	Entrada	HistoryDealGetInteger
DEAL_ENTRY_INOUT	Reversão	HistoryDealGetInteger
DEAL_ENTRY_OUT	Saída	HistoryDealGetInteger
DEAL_MAGIC	Número mágico de uma operação (veja ORDER_MAGIC)	HistoryDealGetInteger
DEAL_ORDER	Número da ordem de uma operação (deal).	HistoryDealGetInteger
DEAL_POSITION_ID	Identificador de uma posição, na abertura, modificação ou alteração da qual esta operação (deal) tomou parte. Cada posição tem um identificador	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	único que é atribuído a todas as operações (deals) executadas para o ativo durante todo o tempo de vida da posição.	
DEAL_PRICE	Preço de uma operação	HistoryDealGetDouble
DEAL_PROFIT	Lucro de uma operação (deal)	HistoryDealGetDouble
DEAL_SWAP	Swap acumulativo no fechamento	HistoryDealGetDouble
DEAL_SYMBOL	Ativo (symbol) de uma operação	HistoryDealGetString
DEAL_TIME	Propriedades de uma Operação (Deal)	HistoryDealGetInteger
DEAL_TIME_MSC	O tempo de execução de	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	negociações em milissegundos desde 01.01.1970	
DEAL_TYPE	Tipo de operação	HistoryDealGetInteger
DEAL_TYPE_BALANCE	Saldo	HistoryDealGetInteger
DEAL_TYPE_BONUS	Bonus	HistoryDealGetInteger
DEAL_TYPE_BUY	Compra	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	Operação de compra cancelada. Pode haver uma situação quando uma operação (deal) de compra executada anteriormente é cancelada. Neste caso, o tipo da operação executada anteriormente (DEAL_TYPE_BUY) é alterada	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	para DEAL_TYPE_BUY_CANCELED, e seu lucro/prejuízo é zerado Lucro/prejuízo obtido anteriormente é cobrado/sacado usando uma operação de saldo separada	
DEAL_TYPE_CHARGE	Cobrança adicional	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Comissão adicional	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Comissão de agente diário	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Comissão de agente mensal	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	Comissão diária	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	Comissão mensal	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	Correção	HistoryDealGetInteger
DEAL_TYPE_CREDIT	Crédito	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
DEAL_TYPE_INTEREST	Taxa de juros	HistoryDealGetInteger
DEAL_TYPE_SELL	Venda	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	Operação de venda cancelada. Pode haver uma situação quando uma operação (deal) de venda executada anteriormente é cancelada. Neste caso, o tipo da operação executada anteriormente (DEAL_TYPE_SELL) é alterada para DEAL_TYPE_SELL_CANCELED, e seu lucro/prejuízo é zerado. Lucro/prejuízo	HistoryDealGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	obtido anteriormente é cobrado/sacado usando uma operação de saldo separada	
DEAL_VOLUME	Volume de uma operação	HistoryDealGetDouble
DRAW_ARROW	Desenha setas	Estilos de Desenho
DRAW_BARS	Exibe como uma seqüência de barras	Estilos de Desenho
DRAW_CANDLES	Exibe como um seqüência de candles	Estilos de Desenho
DRAW_COLOR_ARROW	Desenho de setas multicoloridas	Estilos de Desenho
DRAW_COLOR_BARS	Barras multicoloridas	Estilos de Desenho
DRAW_COLOR_CANDLES	Candles multicoloridos	Estilos de Desenho
DRAW_COLOR_HISTOGRAM	Histograma multicol	Estilos de Desenho

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	orida a partir da linha zero	
DRAW_COLOR_HISTOGRAM2	Histograma multicolorido dos dois buffers do indicador	Estilos de Desenho
DRAW_COLOR_LINE	Linha Multicolorida	Estilos de Desenho
DRAW_COLOR_SECTION	Divisão Multicolorida	Estilos de Desenho
DRAW_COLOR_ZIGZAG	ZigZag Multicolorido	Estilos de Desenho
DRAW_FILLING	Preenchimento de cor entre dos dois níveis	Estilos de Desenho
DRAW_HISTOGRAM	Histograma a partir da linha zero	Estilos de Desenho
DRAW_HISTOGRAM2	Histograma de dois buffers do indicador	Estilos de Desenho
DRAW_LINE	Linha	Estilos de Desenho

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
DRAW_NONE	Não desenhado	Estilos de Desenho
DRAW_SECTION	Seção	Estilos de Desenho
DRAW_ZIGZAG	Estilo Zigzag permite divisão vertical na barra	Estilos de Desenho
ELLIOTT_CYCLE	Ciclo	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	Grande Super-ciclo	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Intermediário	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	Menor	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Minuette	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Minuto	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Primário	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Subminuette	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Super-ciclo	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	Valor vazio em um buffer de indicador	Outras Constantes
ERR_ACCOUNT_WRONG_PROPERTY	Identificador (ID) de propriedade da conta errado	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ERR_ARRAY_BAD_SIZE	Tamanho de array solicitado excede 2 GB	GetLastError
ERR_ARRAY_RESIZE_ERROR	Sem memória suficiente para a realocação de um array, ou uma tentativa de alterar o tamanho de um array estático	GetLastError
ERR_BOOKS_CANNOT_ADD	Profundidade de Mercado não pode ser adicionado	GetLastError
ERR_BOOKS_CANNOT_DELETE	Profundidade de Mercado não pode ser removido	GetLastError
ERR_BOOKS_CANNOT_GET	Os dados da Profundidade de Mercado não podem ser obtidos	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ERR_BOOKS_CANNOT_SUBSCRIBE	Erro em subscrever para receber novos dados da Profundidade de Mercado	GetLastError
ERR_BUFFERS_NO_MEMORY	Sem memória suficiente para a distribuição de buffers de indicador	GetLastError
ERR_BUFFERS_WRONG_INDEX	Índice de buffer de indicador errado	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	Falha ao limpar o diretório (provavelmente um ou mais arquivos estão bloqueados e a operação de remoção falhou)	GetLastError
ERR_CANNOT_DELETE_DIRECTORY	O diretório não pode ser	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	removido	
ERR_CANNOT_DELETE_FILE	Erro excluindo arquivo	GetLastError
ERR_CANNOT_OPEN_FILE	Erro de abertura de arquivo	GetLastError
ERR_CHAR_ARRAY_ONLY	Deve ser um array de tipo char	GetLastError
ERR_CHART_CANNOT_CHANGE	Falha ao alterar ativo e período de um gráfico	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	Falha ao criar timer	GetLastError
ERR_CHART_CANNOT_OPEN	Erro de abertura de gráfico	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	Erro adicionando um indicador no gráfico	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	Erro excluindo um indicador do gráfico	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	Indicador não encontra	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	do no gráfico especificado	
ERR_CHART_NAVIGATE_FAILED	Erro navegando através de gráfico	GetLastError
ERR_CHART_NO_EXPERT	Nenhum Expert Advisor no gráfico que pudesse manipular o evento	GetLastError
ERR_CHART_NO_REPLY	Gráfico não responde	GetLastError
ERR_CHART_NOT_FOUND	Gráfico não encontrado	GetLastError
ERR_CHART_SCREENSHOT_FAILED	Erro criando telas (screenshots)	GetLastError
ERR_CHART_TEMPLATE_FAILED	Erro aplicando o template	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	Sub-janela contendo o indicador não foi	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	encontrada	
ERR_CHART_WRONG_ID	Identificador de gráfico (chart ID) errado	GetLastError
ERR_CHART_WRONG_PARAMETER	Valor de erro do parâmetro para a função que trabalha com gráficos	GetLastError
ERR_CHART_WRONG_PROPERTY	Identificador de propriedade (property ID) do gráfico errado	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	Identificador (ID) de propriedade do indicador customizado errado	GetLastError
ERR_DIRECTORY_NOT_EXIST	Diretório inexistente	GetLastError
ERR_DOUBLE_ARRAY_ONLY	Deve ser um array de tipo double	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ERR_FILE_BINSTRINGSIZE	Tamanho da string deve estar especificado, porque o arquivo está aberto como binário	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	Sem memória suficiente de cache para leitura	GetLastError
ERR_FILE_CANNOT_REWRITE	Arquivo não pode ser reescrito	GetLastError
ERR_FILE_IS_DIRECTORY	Isto não é um arquivo, isto é um diretório	GetLastError
ERR_FILE_ISNOT_DIRECTORY	Isto é um arquivo, não um diretório	GetLastError
ERR_FILE_NOT_EXIST	Arquivo inexistente	GetLastError
ERR_FILE_NOTBIN	O arquivo deve estar aberto como	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	um arquivo binário	
ERR_FILE_NOTCSV	O arquivo deve estar aberto como CSV	GetLastError
ERR_FILE_NOTTOREAD	O arquivo deve estar aberto para leitura	GetLastError
ERR_FILE_NOTTOWRITE	O arquivo deve estar abertura para escrita	GetLastError
ERR_FILE_NOTTXT	O arquivo deve estar aberto como um texto	GetLastError
ERR_FILE_NOTTXTORCSV	O arquivo deve estar aberto como um texto ou CSV	GetLastError
ERR_FILE_READERROR	Erro de leitura	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de arquivo	
ERR_FILE_WRITEERROR	Falha ao escrever um recurso para um arquivo	GetLastError
ERR_FLOAT_ARRAY_ONLY	Deve ser um array de tipo float	GetLastError
ERR_FTP_SEND_FAILED	Envio de arquivo via ftp falhou	GetLastError
ERR_FUNCTION_NOT_ALLOWED	Função de sistema não é permitida para chamar	GetLastError
ERR_GLOBALVARIABLE_EXISTS	Variável global do terminal cliente com o mesmo nome já existe	GetLastError
ERR_GLOBALVARIABLE_NOT_FOUND	Variável global do terminal cliente não foi encontrada	GetLastError
ERR_HISTORY_NOT_FOUND	Histórico solicitado não	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	encontrado	
ERR_HISTORY_WRONG_PROPERTY	Identificador (ID) da propriedade de histórico errado	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Copiando arrays incompatíveis. Array de string pode ser copiado somente para um array de string, e um array numérico - somente em um array numérico	GetLastError
ERR_INCOMPATIBLE_FILE	Um arquivo de texto deve ser usado para arrays de strings, para outros arrays - binários	GetLastError
ERR_INDICATOR_CANNOT_ADD	Erro ao aplicar um	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	indicador ao gráfico	
ERR_INDICATOR_CANNOT_APPLY	O indicador não pode ser aplicado a um outro indicador	GetLastError
ERR_INDICATOR_CANNOT_CREATE	Indicador não pode ser criado	GetLastError
ERR_INDICATOR_CUSTOM_NAME	O primeiro parâmetro no array deve ser o nome do indicador customizado	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	Dado solicitado não encontrado	GetLastError
ERR_INDICATOR_NO_MEMORY	Sem memória suficiente para adicionar o indicador	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Tipo de parâmetro	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	inválido no array ao criar um indicador	
ERR_INDICATOR_PARAMETERS_MISSING	Sem parâmetros ao criar um indicador	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	Ativo desconhecido	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Manuseio de indicador errado	GetLastError
ERR_INDICATOR_WRONG_INDEX	Índice errado de buffer do indicador solicitado	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	Numero errado de parâmetros ao criar um indicador	GetLastError
ERR_INT_ARRAY_ONLY	Deve ser um array de tipo int	GetLastError
ERR_INTERNAL_ERROR	Erro interno	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	inesperado	
ERR_INVALID_ARRAY	Array de um tipo errado, tamanho errado, ou um objeto defeituoso de um array dinâmico	GetLastError
ERR_INVALID_DATETIME	Data e/ou hora inválida	GetLastError
ERR_INVALID_FILEHANDLE	Um arquivo com este handle foi fechado, ou simplesmente não estava aberto	GetLastError
ERR_INVALID_PARAMETER	Parâmetro errado ao chamar a função de sistema	GetLastError
ERR_INVALID_POINTER	Ponteiro errado	GetLastError
ERR_INVALID_POINTER_TYPE	Tipo errado	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de ponteiro	
ERR_LONG_ARRAY_ONLY	Deve ser um array de tipo long	GetLastError
ERR_MAIL_SEND_FAILED	Envio de email falhou	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	Hora do último tick não é conhecida (sem ticks)	GetLastError
ERR_MARKET_NOT_SELECTED	Ativo não está selecionado na janela Observação de Mercado	GetLastError
ERR_MARKET_SELECT_ERROR	Erro adicionando ou excluindo um ativo na janela Observação de Marcado	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	Ativo desconhecido	GetLastError
ERR_MARKET_WRONG_PROPERTY	Identificador de uma propriedade de	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ativo errado	
ERR_MQL5_WRONG_PROPERTY	Identificador de propriedade do programa errado	GetLastError
ERR_NO_STRING_DATE	Sem data na string	GetLastError
ERR_NOT_ENOUGH_MEMORY	Sem memória suficiente para executar a função de sistema	GetLastError
ERR_NOTIFICATION_SEND_FAILED	Falha ao enviar uma notificação	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	Freqüência de envio de notificações em excesso	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	Parâmetro inválido para enviar uma notificação - uma string vazia ou NULL foi passada para a função	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	SendNotification()	
ERR_NOTIFICATION_WRONG_SETTINGS	Configurações de notificações erradas no terminal (ID não está especificado ou permissão não está definida)	GetLastError
ERR_NOTINITIALIZED_STRING	String não inicializada	GetLastError
ERR_NUMBER_ARRAYS_ONLY	Deve ser um array numérico	GetLastError
ERR_OBJECT_ERROR	Erro trabalhando com um objeto gráfico	GetLastError
ERR_OBJECT_GETDATE_FAILED	Não foi possível obter data correspondente ao valor	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Não foi possível obter valor	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	correspondente à data	
ERR_OBJECT_NOT_FOUND	Objeto gráfico não foi encontrado	GetLastError
ERR_OBJECT_WRONG_PROPERTY	Identificador (ID) de uma propriedade de objeto gráfico errado	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Deve ser um array unidimensional	GetLastError
ERR_OPENCL_BUFFER_CREATE	Falha ao criar umbuffer de OpenCL	GetLastError
ERR_OPENCL_CONTEXT_CREATE	Erro criando ocontexto de OpenCL	GetLastError
ERR_OPENCL_EXECUTE	Erro em tempo de execução de programa de OpenCL	GetLastError
ERR_OPENCL_INTERNAL	Erro interno ocorreu aoexecu	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	tar OpenCL	
ERR_OPENCL_INVALID_HANDLE	Handle de OpenCL inválido	GetLastError
ERR_OPENCL_KERNEL_CREATE	Erro criando umkernel de OpenCL	GetLastError
ERR_OPENCL_NOT_SUPPORTED	Funções OpenCLnão são suportados neste computador	GetLastError
ERR_OPENCL_PROGRAM_CREATE	Erro ocorreu aocompilar um program a OpenCL	GetLastError
ERR_OPENCL_QUEUE_CREATE	Falha ao criar um fila de execução em OpenCL	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	Erro ocorreu aoconfigurar parâmetrospara o kernel de OpenCL	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Nome de kernel longo	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	demais(kernel de OpenCL)	
ERR_OPENCL_WRONG_BUFFER_OFFSET	Deslocamento (offset) inválido do buffer de OpenCL	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	Tamanho inválido do buffer de OpenCL	GetLastError
ERR_PLAY_SOUND_FAILED	Reprodução de som falhou	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	Os nomes do recurso dinâmico e do estático equivalem	GetLastError
ERR_RESOURCE_NAME_IS_TOO_LONG	O nome do recurso excede 63 caracteres	GetLastError
ERR_RESOURCE_NOT_FOUND	Recurso com este nome não foi	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	encontrado em EX5	
ERR_RESOURCE_UNSUPPORTED_TYPE	Tipo de recurso não suportado ou seu tamanho excede 16 Mb	GetLastError
ERR_SERIES_ARRAY	Série de tempo (timeseries) não pode ser usada	GetLastError
ERR_SHORT_ARRAY_ONLY	Deve ser um array de tipo short	GetLastError
ERR_SMALL_ARRAY	Array pequeno demais, a posição inicial está fora do array	GetLastError
ERR_SMALL_ASSERIES_ARRAY	O array de recepção está declarado como AS_SERIES, e é de tamanho insuficiente.	GetLastError
ERR_STRING_OUT_OF_MEMORY	Sem memória	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	suficiente para a string	
ERR_STRING_RESIZE_ERROR	Sem memória suficiente para a realocação de string	GetLastError
ERR_STRING_SMALL_LEN	O comprimento da string é menor que o esperado	GetLastError
ERR_STRING_TIME_ERROR	Erro convertendo string em data	GetLastError
ERR_STRING_TOO_BIGNUMBER	Número grande demais, maior que ULONG_MAX	GetLastError
ERR_STRING_UNKNOWNTYPE	Tipo de dado desconhecido ao converter para uma string	GetLastError
ERR_STRING_ZEROADDED	0 adicionado ao final da string, uma	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	operação inútil	
ERR_STRINGPOS_OUTOFRANGE	Posição fora da string	GetLastError
ERR_STRUCT_WITHOBJECTS_ORCLASS	A estrutura contém objetos de strings e/ou arrays dinâmicos e/ou estrutura de tais objetos e/ou classes	GetLastError
ERR_SUCCESS	A operação concluída com sucesso	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	Identificador de propriedade do terminal errado	GetLastError
ERR_TOO_LONG_FILENAME	Nome de arquivo longo demais	GetLastError
ERR_TOO_MANY_FILES	Mais que 64 arquivos não podem ser abertos	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ao mesmo tempo	
ERR_TOO_MANY_FORMATTERS	Quantidade de especificadores de formato maior que de parâmetros	GetLastError
ERR_TOO_MANY_PARAMETERS	Quantidade de parâmetros maior que de especificadores de formato	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	Operação (deal) não encontrada	GetLastError
ERR_TRADE_DISABLED	Negociação via Expert Advisors proibida	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	Ordem não encontrada	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	Posição não encontrada	GetLastError
ERR_TRADE_SEND_FAILED	Envio de solicitaç	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ão de negociação falhou	
ERR_TRADE_WRONG_PROPERTY	Identificador (ID) de propriedade da negociação (trade) errado	GetLastError
ERR_USER_ERROR_FIRST	Erros definidos pelo usuário começam com este código	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	Failed to connect to specified URL	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	Invalid URL	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	HTTP request failed	GetLastError
ERR_WEBREQUEST_TIMEOUT	Timeout exceeded	GetLastError
ERR_WRONG_DIRECTORYNAME	Nome de diretório errado	GetLastError
ERR_WRONG_FILEHANDLE	Handle de arquivo errado	GetLastError

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ERR_WRONG_FILENAME	Nome de arquivo inválido	GetLastError
ERR_WRONG_FORMATSTRING	string de formato inválido	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Parâmetro errado na chamada interna da função do terminal cliente	GetLastError
ERR_WRONG_STRING_DATE	Data errada na string	GetLastError
ERR_WRONG_STRING_OBJECT	Objeto de string defeituoso	GetLastError
ERR_WRONG_STRING_PARAMETER	Parâmetro de tipo string defeituoso	GetLastError
ERR_WRONG_STRING_TIME	Hora errada na string	GetLastError
ERR_ZEROSIZE_ARRAY	Um array de comprimento zero	GetLastError
FILE_ACCESS_DATE	Data do último acesso ao arquivo	FileGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
FILE_ANSI	Strings de tipo ANSI (símbolos de um byte). Flag é usado em FileOpen()	FileOpen
FILE_BIN	Modo de leitura/escrita binária (sem conversão string para string). Flag é usado em FileOpen()	FileOpen
FILE_COMMON	O caminho do arquivo na pasta comum de todos os terminais cliente \Terminal\Common\Files. Flag é usado em funções FileOpen(), FileCopy(),	FileOpen, FileCopy, FileMove, FileIsExist

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	FileMove() e em FileIsExist().	
FILE_CREATE_DATE	Data de criação	FileGetInteger
FILE_CSV	Arquivo CSV (todos seus elementos são convertidos para strings do tipo apropriado, unicode ou ansi, e separados por separador). Flag é usado em FileOpen()	FileOpen
FILE_END	Obtém o sinal de término de arquivo	FileGetInteger
FILE_EXISTS	Verifica a existência	FileGetInteger
FILE_IS_ANSI	O arquivo está aberto como ANSI (veja	FileGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	FILE_ANSI)	
FILE_IS_BINARY	O arquivo está aberto como um arquivo binário (veja FILE_BIN)	FileGetInteger
FILE_IS_COMMON	O arquivo está aberto em uma pasta compartilhada por todos os terminais (veja FILE_COMMON)	FileGetInteger
FILE_IS_CSV	O arquivo está aberto como CSV (veja FILE_CSV)	FileGetInteger
FILE_IS_READABLE	O arquivo aberto pode ser lido (veja FILE_READ)	FileGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
FILE_IS_TEXT	O arquivo está aberto como um arquivo de texto (veja FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	O arquivo aberto pode ser escrito (veja FILE_WRITE)	FileGetInteger
FILE_LINE_END	Obtém o sinal de termino de linha	FileGetInteger
FILE_MODIFY_DATE	Data da última modificação	FileGetInteger
FILE_POSITION	Posição de um ponteiro no arquivo	FileGetInteger
FILE_READ	Arquivo é aberto para leitura. Flag é usado em FileOpen(). Ao abrir um arquivo, a especific	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ação de FILE_WRITE e/ou FILE_READ é requerida.	
FILE_REWRITE	Possibilidade de reescrever o arquivo usando as funções FileCopy() e FileMove(). O arquivo deve existir ou deve estar aberto para escrita, de outra forma o arquivo não será aberto.	FileCopy, FileMove
FILE_SHARE_READ	Acesso compartilhado para leitura a partir de vários programas. Flag é usado em FileOpen(), mas ele não	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	substitui a necessidade de indicar o flag FILE_WRITE e/ou FILE_READ ao abrir um arquivo.	
FILE_SHARE_WRITE	Acesso compartilhado para escrita a partir de vários programas. Flag é usado em FileOpen(), mas ele não substitui a necessidade de indicar o flag FILE_WRITE e/ou FILE_READ ao abrir um arquivo.	FileOpen
FILE_SIZE	Tamanho de arquivo em bytes	FileGetInteger
FILE_TXT	Arquivo de texto simples	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	(semelhando ao arquivo csv, mas sem levar em conta os separadores). Flag é usado em FileOpen()	
FILE_UNICODE	Strings de tipo UNICODE (símbolos de dois bytes). Flag é usado em FileOpen()	FileOpen
FILE_WRITE	Arquivo é aberto para escrita. Flag é usado em FileOpen(). Ao abrir um arquivo, a especificação de FILE_WRITE e/ou FILE_READ é requerida.	FileOpen

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
FLT_DIG	Número de algarismos decimais significativos para o tipo float	Constantes de Tipo Numérico
FLT_EPSILON	Valor mínimo, que satisfaz a condição: 1.0+DBL_EPSILON != 1.0 (para o tipo float)	Constantes de Tipo Numérico
FLT_MANT_DIG	Quantidade de bits em uma mantissa para o tipo float	Constantes de Tipo Numérico
FLT_MAX	Valor máximo, que pode ser representado pelo tipo float	Constantes de Tipo Numérico
FLT_MAX_10_EXP	Valor decimal máximo de grau de expoent	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	e para o tipo float	
FLT_MAX_EXP	Valor binário máximo de grau de expoente para o tipo float	Constantes de Tipo Numérico
FLT_MIN	Valor positivo mínimo, que pode ser representado pelo tipo float	Constantes de Tipo Numérico
FLT_MIN_10_EXP	Valor decimal mínimo de grau de expoente para o tipo float	Constantes de Tipo Numérico
FLT_MIN_EXP	Valor binário mínimo de grau de expoente para o tipo float	Constantes de Tipo Numérico
FRIDAY	Sexta-feira	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	Linha correspo	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ndente à tendência de baixa	
GANN_UP_TREND	Linha correspondente à tendência de alta	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	Linha Jaw	Linhas de Indicadores
GATORLIPS_LINE	Linha Lips	Linhas de Indicadores
GATORTEETH_LINE	Linha Teeth	Linhas de Indicadores
IDABORT	Botão "Abort" foi pressionado	MessageBox
IDCANCEL	Botão "Cancel" foi pressionado	MessageBox
IDCONTINUE	Botão "Continue" foi pressionado	MessageBox
IDIGNORE	Botão "Ignore" foi pressionado	MessageBox
IDNO	Botão "No" foi pressionado	MessageBox
IDOK	Botão "OK" foi	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	pressionado	
IDRETRY	Botão "Retry" foi pressionado	MessageBox
IDTRYAGAIN	Botão "Try Again" foi pressionado	MessageBox
IDYES	Botão "Yes" foi pressionado	MessageBox
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity Channel Index	IndicatorCreate, IndicatorParameters
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters
IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillato	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	r	
IND_ICHIMOKU	Ichimoku Kinko Hyo	IndicatorCreate, IndicatorParameters
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_TRIX	Triple Exponential	IndicatorCreate, IndicatorParameters

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	Moving Averages Oscillator	
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	Buffers auxiliares para cálculos intermediários	SetIndexBuffer
INDICATOR_COLOR_INDEX	Cor	SetIndexBuffer
INDICATOR_DATA	Dados para desenho	SetIndexBuffer
INDICATOR_DIGITS	Acuracidade de desenho de valores de indicador	IndicatorSetInteger
INDICATOR_HEIGHT	Altura fixa da janela do indicador (o comando de pré-processador #property	IndicatorSetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	indicator_height)	
INDICATOR_LEVELCOLOR	Cor da linha de nível	IndicatorSetInteger
INDICATOR_LEVELS	Número de níveis na janela do indicador	IndicatorSetInteger
INDICATOR_LEVELSTYLE	Estilo da linha de nível	IndicatorSetInteger
INDICATOR_LEVELTEXT	Nível de descrição	IndicatorSetString
INDICATOR_LEVELVALUE	Valor de nível	IndicatorSetDouble
INDICATOR_LEVELWIDTH	Espessura da linha de nível	IndicatorSetInteger
INDICATOR_MAXIMUM	Máximo da janela do indicador	IndicatorSetDouble
INDICATOR_MINIMUM	Mínimo da janela do indicador	IndicatorSetDouble
INDICATOR_SHORTNAME	Nome do indicador abreviado	IndicatorSetString

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
INT_MAX	Valor máximo, que pode ser representado pelo tipo int	Constantes de Tipo Numérico
INT_MIN	Valor mínimo, que pode ser representado pelo tipo int	Constantes de Tipo Numérico
INVALID_HANDLE	Manipulador incorreto	Outras Constantes
IS_DEBUG_MODE	Flag que indica que um programa MQL5 opera em modo de depuração	Outras Constantes
IS_PROFILE_MODE	Flag que indica que um programa MQL5 opera em modo de profiling	Outras Constantes
KIJUNSEN_LINE	Linha Kijun-sen	Linhas de Indicadores
LICENSE_DEMO	Um versão	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	trial de um produto pago do Market. Funciona somente no Provador de Estratégia	
LICENSE_FREE	Um versão gratuita ilimitada	MQLInfoInteger
LICENSE_FULL	Uma versão licenciada adquirida permite pelo menos 5 ativações. O vendedor pode aumentar o número permitido de ativações.	MQLInfoInteger
LICENSE_TIME	Uma versão com termo de licença limitada	MQLInfoInteger
LONG_MAX	Valor máximo,	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	que pode ser representado pelo tipo long	
LONG_MIN	Valor mínimo, que pode ser representado pelo tipo long	Constantes de Tipo Numérico
LOWER_BAND	Limite inferior	Linhas de Indicadores
LOWER_HISTOGRAM	Histograma inferior	Linhas de Indicadores
LOWER_LINE	Linha de baixo	Linhas de Indicadores
M_1_PI	1/pi	Constantes Matemáticas
M_2_PI	2/pi	Constantes Matemáticas
M_2_SQRTPI	2/sqrt(pi)	Constantes Matemáticas
M_E	e	Constantes Matemáticas
M_LN10	ln(10)	Constantes Matemáticas
M_LN2	ln(2)	Constantes Matemáticas
M_LOG10E	log10(e)	Constantes Matemáticas
M_LOG2E	log2(e)	Constantes Matemáticas
M_PI	pi	Constantes Matemáticas
M_PI_2	pi/2	Constantes Matemáticas
M_PI_4	pi/4	Constantes Matemáticas
M_SQRT1_2	1/sqrt(2)	Constantes Matemáticas
M_SQRT2	sqrt(2)	Constantes Matemáticas

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
MAIN_LINE	Linha principal	Linhas de Indicadores
MB_ABORTRETRYIGNORE	Janela de mensagem contém três botões: Abort, Retry e Ignore	MessageBox
MB_CANCELTRYCONTINUE	Janela de mensagem contém três botões: Cancel, Try Again, Continue	MessageBox
MB_DEFBUTTON1	O primeiro botão MB_DEFBUTTON1 - é o default, se os outros botões MB_DEFBUTTON2, MB_DEFBUTTON3, ou MB_DEFBUTTON4 não forem especificados	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
MB_DEFBUTTON2	O segundo botão é o default	MessageBox
MB_DEFBUTTON3	O terceiro botão é o default	MessageBox
MB_DEFBUTTON4	O quarto botão é o default	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	O ícone de sinal de exclamação/aviso	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	O sinal icercado	MessageBox
MB_ICONQUESTION	O ícone de sinal de interrogação	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	O ícone de sinal STOP	MessageBox
MB_OK	Janela de mensagem contém somente um botão: OK. Default	MessageBox
MB_OKCANCEL	Janela de mensagem contém dois	MessageBox

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	botões: OK e Cancel	
MB_RETRYCANCEL	Janela de mensagem contém dois botões: Retry e Cancel	MessageBox
MB_YESNO	Janela de mensagem contém dois botões: Yes e No	MessageBox
MB_YESNOCANCEL	Janela de mensagem contém três botões: Yes, No e Cancel	MessageBox
MINUSDI_LINE	Linha -DI	Linhas de Indicadores
MODE_EMA	Média exponencial	Métodos de Suavização
MODE_LWMA	Média linear-ponderada	Métodos de Suavização
MODE_SMA	Média simples	Métodos de Suavização
MODE_SMMA	Média suavizada	Métodos de Suavização

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
MONDAY	Segunda-feira	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	O flag, que indica o modo de depuração	MQLInfoInteger
MQL_DLLS_ALLOWED	A permissão para usar DLL para o dado programa executado	MQLInfoInteger
MQL_FRAME_MODE	O sinalizador, indicando o que o Expert Advisor está operando na coleta de resultado no modo de otimização dos quadros	MQLInfoInteger
MQL_LICENSE_TYPE	Tipo de licença do módulo EX5. A licença refere-se ao	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	módulo EX5, a partir do qual é feita uma solicitação usando MQLInfoInteger(MQL_LICENSE_TYPE).	
MQL_MEMORY_LIMIT	Maximum possible amount of dynamic memory for MQL5 program in MB	MQLInfoInteger
MQL_MEMORY_USED	The memory size used by MQL5 program in MB	MQLInfoInteger
MQL_OPTIMIZATION	O flag, que indica o processo de otimização	MQLInfoInteger
MQL_PROFILER	O flag, que indica que o programa está	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	operando em modo de profiling de código	
MQL_PROGRAM_NAME	Nome do programa MQL5 executado	MQLInfoString
MQL_PROGRAM_PATH	Caminho para o dado programa executado	MQLInfoString
MQL_PROGRAM_TYPE	Tipo do programa MQL5	MQLInfoInteger
MQL_SIGNALS_ALLOWED	The permission to modify the Signals for the given executed program	MQLInfoInteger
MQL_TESTER	O flag, que indica o processo do testador	MQLInfoInteger
MQL_TRADE_ALLOWED	A permissão para negociar para o dado	MQLInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	programa executado	
MQL_VISUAL_MODE	O flag, que indica o processo de testador visual	MQLInfoInteger
NULL	Zero para quaisquer tipos	Outras Constantes
OBJ_ALL_PERIODS	O objeto é desenhado em todas das janelas de tempo	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	Seta	Tipos de Objeto
OBJ_ARROW_BUY	Comprar	Tipos de Objeto
OBJ_ARROW_CHECK	Conferido	Tipos de Objeto
OBJ_ARROW_DOWN	Seta para baixo	Tipos de Objeto
OBJ_ARROW_LEFT_PRICE	Etiqueta de Preço à Esquerda	Tipos de Objeto
OBJ_ARROW_RIGHT_PRICE	Rótulo de Preço Direito	Tipos de Objeto
OBJ_ARROW_SELL	Vender	Tipos de Objeto

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
OBJ_ARROW_STOP	Sinal Stop	Tipos de Objeto
OBJ_ARROW_THUMB_DOWN	Polegar para Baixo	Tipos de Objeto
OBJ_ARROW_THUMB_UP	Polegares para cima	Tipos de Objeto
OBJ_ARROW_UP	Seta para cima	Tipos de Objeto
OBJ_ARROWED_LINE	Linha de Seta	Tipos de Objeto
OBJ_BITMAP	Bitmap	Tipos de Objeto
OBJ_BITMAP_LABEL	Rótulo Bitmap	Tipos de Objeto
OBJ_BUTTON	Botão	Tipos de Objeto
OBJ_CHANNEL	Canal Eqüidistante	Tipos de Objeto
OBJ_CHART	Gráfico	Tipos de Objeto
OBJ_CYCLES	Linhas de Ciclo	Tipos de Objeto
OBJ_EDIT	Edição	Tipos de Objeto
OBJ_ELLIOTWAVE3	Onde de Correção de Elliott	Tipos de Objeto
OBJ_ELLIOTWAVE5	Onda Motriz de Elliott	Tipos de Objeto
OBJ_ELLIPSE	Elipse	Tipos de Objeto
OBJ_EVENT	O objeto "Evento" correspondendo a um evento no	Tipos de Objeto

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	calendário econômico	
OBJ_EXPANSION	Expansão de Fibonacci	Tipos de Objeto
OBJ_FIBO	Retração de Fibonacci	Tipos de Objeto
OBJ_FIBOARC	Arcos de Fibonacci	Tipos de Objeto
OBJ_FIBOCHANNEL	Canal de Fibonacci	Tipos de Objeto
OBJ_FIBOFAN	Fibonacci Fan	Tipos de Objeto
OBJ_FIBOTIMES	Fusos Horários de Fibonacci	Tipos de Objeto
OBJ_GANNFAN	Ventilador Gann	Tipos de Objeto
OBJ_GANNGRID	Grade Gann	Tipos de Objeto
OBJ_GANNLINE	Linha Gann	Tipos de Objeto
OBJ_HLINE	Linha Horizontal	Tipos de Objeto
OBJ_LABEL	Rótulo	Tipos de Objeto
OBJ_NO_PERIODS	O objeto não desenhado em nenhum	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	a janela de tempo	
OBJ_PERIOD_D1	O objeto é desenhado em gráficos diários	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	O objeto é desenhado em gráficos de 1 hora	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	O objeto é desenhado em gráficos de 12 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	O objeto é desenhado em gráficos de 2 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	O objeto é desenhado em gráficos de 3 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	O objeto é desenhado em gráficos de 4 horas	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
OBJ_PERIOD_H6	O objeto é desenhado em gráficos de 6 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	O objeto é desenhado em gráficos de 8 horas	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	O objeto é desenhado em gráficos de 1 minuto	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	O objeto é desenhado em gráficos de 10 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	O objeto é desenhado em gráficos de 12 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	O objeto é desenhado em gráficos de 15 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	O objeto é	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	desenhado em gráficos de 2 minutos	
OBJ_PERIOD_M20	O objeto é desenhado em gráficos de 20 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	O objeto é desenhado em gráficos de 3 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	O objeto é desenhado em gráficos de 30 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	O objeto é desenhado em gráficos de 4 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	O objeto é desenhado em gráficos de 5 minutos	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	O objeto é desenhado em	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	gráficos de 6 minutos	
OBJ_PERIOD_MN1	O objeto é desenhado em gráficos mensais	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	O objeto é desenhado em gráficos semanais	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	Forquilha de Andrews	Tipos de Objeto
OBJ_RECTANGLE	Retângulo	Tipos de Objeto
OBJ_RECTANGLE_LABEL	O objeto "Rótulo de Retângulo" para criação e design de interface gráfico customizada.	Tipos de Objeto
OBJ_REGRESSION	Canal de Regressão Linear	Tipos de Objeto
OBJ_STDDEVCHANNEL	Canal de Desvio Padrão	Tipos de Objeto
OBJ_TEXT	Texto	Tipos de Objeto

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
OBJ_TREND	Linha de Tendência	Tipos de Objeto
OBJ_TRENDBYANGLE	Linha de Tendência por Ângulo	Tipos de Objeto
OBJ_TRIANGLE	Triângulo	Tipos de Objeto
OBJ_VLINE	Linha Vertical	Tipos de Objeto
OBJPROP_ALIGN	Alinhamento de texto horizontal no objeto "Edit" (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	Localização do ponto de ancoragem de um objeto gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	Ângulo. Para os objetos sem ângulo especificado, criado a partir de um programa, o valor é igual a EMPTY_VALUE	ObjectSetDouble, ObjectGetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
OBJPROP_ARROWCODE	Código de seta para o objeto Seta	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	Objeto no pano de fundo	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	A cor de fundo para OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	O nome do arquivo BMP para rótulo Bitmap. Veja também Recursos	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	Cor de borda para os objetos OBJ_EDIT e OBJ_BUTTON	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	Tipo de borda para o objeto "Rectangle label"	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	ID do objeto	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	"Chart" (OBJ_CHART). Permite trabalhar com as propriedades deste objeto como um gráfico normal usando as funções descritas em Operações de Gráfico, mas existem algumas exceções.	
OBJPROP_CHART_SCALE	A escala para o objeto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	Cor	ObjectSetInteger, ObjectGetInteger
OBJPROP_CORNER	O canto do gráfico para linkar um objeto gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Hora de criação do objeto	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
OBJPROP_DATE_SCALE	Exibição da escala de tempo para o objeto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Nível de marcação da Onda de Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	Desvio para o Canal de Desvio Padrão	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	Tendência do objeto Gann	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	Exibição de linhas para marcação da Onda de Elliott	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	Exibição da elipse completa do objeto Arco de Fibonacci (OBJ_FIBOARC)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Preenche um objeto com cor (para	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_FONT	Font	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	Tamanho de fonte	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	Proíbe exibir o nome de um objeto gráfico na lista de objetos do menu do terminal "Gráficos" - "Objetos" - "Lista de objetos". O valor true permite ocultar um objeto da lista. Por default, true é	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	definido para os objetos que exibem eventos de calendário, histórico de negociação e para objetos criados a partir de programas MQL5. Para ver tais objetos gráficos e acessar suas propriedades, clique no botão "TODOS" na janela "Lista de objetos".	
OBJPROP_LEVELCOLOR	Cor das linhas de nível	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Número de níveis	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Estilo da linha de	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	nível	
OBJPROP_LEVELTEXT	Nível de descrição	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	Valor de nível	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	Espessura do nível de linha	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Nome de objeto	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	Janela de tempo para o objeto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	Coordenada de Preço	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Exibição da escala de preço para o objeto Chart	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Uma linha vertical indo através de todas as janelas de um gráfico	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Raios indo para	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	esquerda	
OBJPROP_RAY_RIGHT	Raios indo para direita	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	Capacidade de editar texto no objeto Edit	ObjectSetInteger, ObjectGetInteger
OBJPROP_SCALE	Escala (propriedades de objetos Gann e Arcos de Fibonnacci)	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	Disponibilidade de objeto	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	Objeto é selecionado	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	Estado de botão (pressionado / não pressionado)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Estilo	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Ativo para o objeto Chart	ObjectSetString, ObjectGetString
OBJPROP_TEXT	Descrição do objeto	ObjectSetString, ObjectGetString

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	(o texto contido no objeto)	
OBJPROP_TIME	Coordenada de tempo	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	Visibilidade de um objeto nas janelas de tempo	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	O texto de um "tooltip" (dica). Se a propriedade não é definida, então o "tooltip" gerado automaticamente pelo terminal é exibido. Um "tooltip" pode ser desabilitado através da atribuição do valor "\n" (quebra de	ObjectSetString, ObjectGetString

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	linha) a ele	
OBJPROP_TYPE	Tipo de objeto	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	Espessura de linha	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	A distância em pixels ao longo do eixo X a partir do canto de vinculação (see note)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	A coordenada X do canto superior esquerdo da área visível retangular nos objetos gráficos "Bitmap Label" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). O valor é definido em pixels em relação	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ao canto superior esquerdo da imagem original.	
OBJPROP_XSIZE	O tamanho do objeto Chart ao longo do eixo X (largura em pixels) Especificado para objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	ObjectSetInteger, ObjectGetInteger
OBJPROP_YDISTANCE	A distância em pixels ao longo do eixo Y a partir do canto de vinculaç	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ão (see note)	
OBJPROP_YOFFSET	A coordenada X do canto superior esquerdo da área visível retangular nos objetos gráficos "Bitmap Label" e "Bitmap" (OBJ_BITMAP_LABEL e OBJ_BITMAP). O valor é definido em pixels em relação ao canto superior esquerdo da imagem original.	ObjectSetInteger, ObjectGetInteger
OBJPROP_YSIZE	O tamanho do objeto Chart ao longo do eixo X (largura em pixels) Especific	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ado para objetos OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL.	
OBJPROP_ZORDER	Prioridade de um objeto gráfico para receber eventos de clickagem em um gráfico (CHARTEVENT_CLICK). O valor zero default é definido ao se criar um objeto; a prioridade pode ser aumenta	ObjectSetInteger, ObjectGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	da se necessário. Ao colocar objetos um sobre o outro, somente aquele com a maior prioridade receberá o evento CHARTEVENT_CLICK.	
ORDER_COMMENT	Comentário sobre a ordem	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Esta política de preenchimento significa que uma ordem pode ser preenchida somente na quantidade especificada. Se a quantidade desejada do ativo não está	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	disponível no mercado, a ordem não será executada. O volume requerido pode ser preenchido usando várias ofertas disponíveis no mercado no momento.	
ORDER_FILLING_IOC	Este modo significa que um negociador concorda em executar uma operação com o volume máximo disponível no mercado conforme indicado na ordem. No caso do	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	volume integral de uma ordem não puder ser preenchido, o volume disponível dele será preenchido, e o volume restante será cancelado.	
ORDER_FILLING_RETURN	Esta política é usada somente para ordens a mercado (ORDER_TYPE_BUY e ORDER_TYPE_SELL), ordens limit e stop limit (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	LIMIT e ORDER_TYPE_SELL_STOP_LIMIT) e somente para os ativos com execução a Mercado ou execução em um sistema de negociação externo (Exchange)***. No caso de um preenchimento parcial, uma ordem a mercado ou do tipo limit com volume remanescente não é cancelada, mas processada posteriormente. Para a ativação das	

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ordens ORDER_TYPE_BUY_STOP_LIMIT e ORDER_TYPE_SELL_STOP_LIMIT, uma ordem limit correspondente, ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT com o tipo de execução ORDER_FILLING_RETURN, é criada.	
ORDER_MAGIC	ID de um Expert Advisor que colocou a ordem (projetado para garantir que cada Expert Advisor coloque seu próprio número único)	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ORDER_POSITION_ID	Identificador de posição que é definido para uma ordem tão logo ela é executada. Cada ordem executada resulta em uma operação que abre ou modifica uma posição já existente. O identificador desta exata posição é atribuída à ordem executada neste momento.	OrderGetInteger, HistoryOrderGetInteger
ORDER_PRICE_CURRENT	O preço corrente do ativo de uma ordem	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Preço especific	OrderGetDouble, HistoryOrderGetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ado na ordem	
ORDER_PRICE_STOPLIMIT	O preço de ordem Limit para uma ordem StopLimit	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	Valor de Stop Loss	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Estado de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	Ordem cancelada pelo cliente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	Ordem expirada	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	Ordem executada completamente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	Ordem executada parcialmente	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	Ordem aceita	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	Ordem rejeitada	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	Ordem está sendo	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	registrada (aplicação para o sistema de negociação)	
ORDER_STATE_REQUEST_CANCEL	Ordem está sendo excluída (excluindo a partir do sistema de negociação)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	Ordem está sendo modificada (alterando seus parâmetros)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	Ordem verificada, mas ainda não aceita pela corretora (broker)	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	Ativo (symbol) de uma ordem	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	Ordem válida até o	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	final do dia corrente de negociação	
ORDER_TIME_DONE	Hora de execução ou cancelamento de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	Tempo de execução e cancelamento de ordens em milissegundos desde 01.01.1970	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Hora de expiração de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	Ordem válida até cancelamento	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Hora de configuração de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	O tempo para	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	colocar uma ordem de execução em milissegundos desde 01.01.1970	
ORDER_TIME_SPECIFIED	Ordem válida até expiração	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	A ordem permanecerá efetiva até 23:59:59 do dia especificado. Se esta hora está fora de uma sessão de negociação, a ordem expira na hora de negociação mais próxima.	OrderGetInteger, HistoryOrderGetInteger
ORDER_TP	Valor de Take Profit	OrderGetDouble, HistoryOrderGetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ORDER_TYPE	Tipo de ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	Ordem de Comprar a Mercado	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	Ordem pendente Buy Limit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	Ordem pendente Buy Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Ao alcançar o preço da ordem, uma ordem pendente Buy Limit é colocada no preço StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_FILLING	Type de preenchimento de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	Ordem de Vender a Mercado	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	Ordem pendente Sell Limit	OrderGetInteger, HistoryOrderGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ORDER_TYPE_SELL_STOP	Ordem pendente Sell Stop	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	Ao alcançar o preço da ordem, uma ordem pendente Sell Limit é colocada no preço StopLimit	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	tempo de duração de uma ordem	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Volume corrente de uma ordem	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Volume inicial de uma ordem	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	Janela de tempo corrente	Janela temporal de Gráfico
PERIOD_D1	1 dia	Janela temporal de Gráfico
PERIOD_H1	1 hora	Janela temporal de Gráfico
PERIOD_H12	12 horas	Janela temporal de Gráfico
PERIOD_H2	2 horas	Janela temporal de Gráfico
PERIOD_H3	3 horas	Janela temporal de Gráfico
PERIOD_H4	4 horas	Janela temporal de Gráfico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
PERIOD_H6	6 horas	Janela temporal de Gráfico
PERIOD_H8	8 horas	Janela temporal de Gráfico
PERIOD_M1	1 minuto	Janela temporal de Gráfico
PERIOD_M10	10 minutos	Janela temporal de Gráfico
PERIOD_M12	12 minutos	Janela temporal de Gráfico
PERIOD_M15	15 minutos	Janela temporal de Gráfico
PERIOD_M2	2 minutos	Janela temporal de Gráfico
PERIOD_M20	20 minutos	Janela temporal de Gráfico
PERIOD_M3	3 minutos	Janela temporal de Gráfico
PERIOD_M30	30 minutos	Janela temporal de Gráfico
PERIOD_M4	4 minutos	Janela temporal de Gráfico
PERIOD_M5	5 minutos	Janela temporal de Gráfico
PERIOD_M6	6 minutos	Janela temporal de Gráfico
PERIOD_MN1	1 mês	Janela temporal de Gráfico
PERIOD_W1	1 semana	Janela temporal de Gráfico
PLOT_ARROW	Código de seta para estilo DRAW_ARROW	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	Deslocamento vertical de setas para estilo	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	DRAW_ARROW	
PLOT_COLOR_INDEXES	O número de cores	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Número de barras iniciais sem desenho e valores na Janela de Dados	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	Tipo de construção gráfica	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Um valor vazio para plotagem, para a qual não há desenho	PlotIndexSetDouble
PLOT_LABEL	O nome da série gráfica do indicador para exibir na Janela de Dados Ao trabalhar com estilos gráficos complex	PlotIndexSetString

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	os exigindo vários buffers de indicador para a exposição, os nomes para cada buffer pode ser especificado utilizando ";" como separador. O código de exemplo é mostrado na DRAW_CANDLES	
PLOT_LINE_COLOR	O índice de um buffer contendo a cor de desenho	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Estilo de linha de desenho	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	A espessura da linha de desenho	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
PLOT_SHIFT	Deslocamento da plotagem do indicador ao longo do eixo tempo em barras	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHOW_DATA	Sinal de exibição de valores de construção na Janela de Dados	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Linha +DI	Linhas de Indicadores
POINTER_AUTOMATIC	Ponteiro de qualquer objeto criado automaticamente (sem usar new())	CheckPointer
POINTER_DYNAMIC	Ponteiro do objeto criado por meio do operador new()	CheckPointer
POINTER_INVALID	Ponteiro incorreto	CheckPointer

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
POSITION_COMMENT	Comentário de uma posição	PositionGetString
POSITION_COMMISSION	Comissão	PositionGetDouble
POSITION_IDENTIFIER	Identificador de posição é um número único que é atribuído para toda nova posição aberta e não se altera durante todo o tempo de vida da posição. Movimentações de uma posição não alteram seu identificador.	PositionGetInteger
POSITION_MAGIC	Número mágico de uma posição (veja ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	Preço corrente	PositionGetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	do ativo de uma posição	
POSITION_PRICE_OPEN	Preço de abertura de uma posição	PositionGetDouble
POSITION_PROFIT	Lucro corrente	PositionGetDouble
POSITION_SL	Nível de Stop Loss de uma posição aberta	PositionGetDouble
POSITION_SWAP	Swap acumulativo	PositionGetDouble
POSITION_SYMBOL	Ativo (symbol) da posição	PositionGetString
POSITION_TIME	Hora de abertura de uma posição	PositionGetInteger
POSITION_TIME_MSC	Posição de tempo de abertura em milissegundos desde 01.01.1970	PositionGetInteger
POSITION_TIME_UPDATE	Posição de tempo de alteração em	PositionGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	segundos desde 01.01.1970	
POSITION_TIME_UPDATE_MSC	Posição de tempo de alteração em milissegundo desde 01.01.1970	PositionGetInteger
POSITION_TP	Nível de Take Profit de uma posição aberta	PositionGetDouble
POSITION_TYPE	Tipo de posição	PositionGetInteger
POSITION_TYPE_BUY	Compra	PositionGetInteger
POSITION_TYPE_SELL	Venda	PositionGetInteger
POSITION_VOLUME	Volume de uma posição	PositionGetDouble
PRICE_CLOSE	Preço de Fechamento	Constantes de Preço
PRICE_HIGH	O preço máximo para o período	Constantes de Preço
PRICE_LOW	O preço mínimo para o período	Constantes de Preço
PRICE_MEDIAN	O preço mediano	Constantes de Preço

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	, (máximo + mínimo) /2	
PRICE_OPEN	Preço de Abertura	Constantes de Preço
PRICE_TYPICAL	Preço típico, (máximo + mínimo + fechamento)/3	Constantes de Preço
PRICE_WEIGHTED	Preço médio, (máximo + mínimo + fechamento + fechamento)/4	Constantes de Preço
PROGRAM_EXPERT	Expert	MQLInfoInteger
PROGRAM_INDICATOR	Indicador	MQLInfoInteger
PROGRAM_SCRIPT	Script	MQLInfoInteger
REASON_ACCOUNT	Another account has been activated or reconnection to the trade server has occurred due to changes	UninitializeReason, OnDeinit

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	in the account settings	
REASON_CHARTCHANGE	Symbol or chart period has been changed	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	Chart has been closed	UninitializeReason, OnDeinit
REASON_CLOSE	Terminal has been closed	UninitializeReason, OnDeinit
REASON_INITFAILED	This value means that OnInit() handler has returned a nonzero value	UninitializeReason, OnDeinit
REASON_PARAMETERS	Input parameters have been changed by a user	UninitializeReason, OnDeinit
REASON_PROGRAM	Expert Advisor terminated its operation by calling the ExpertRemove() function	UninitializeReason, OnDeinit

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
REASON_RECOMPILE	Program has been recompiled	UninitializeReason, OnDeinit
REASON_REMOVE	Program has been deleted from the chart	UninitializeReason, OnDeinit
REASON_TEMPLATE	A new template has been applied	UninitializeReason, OnDeinit
SATURDAY	Sábado	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	Posição corrente de um ponteiro de arquivo	FileSeek
SEEK_END	Fim de arquivo	FileSeek
SEEK_SET	Início de arquivo	FileSeek
SENKOUSPANA_LINE	Linha Senkou Span A	Linhas de Indicadores
SENKOUSPANB_LINE	Linha Senkou Span B	Linhas de Indicadores
SERIES_BARS_COUNT	Quantidade de barras para o ativo-período para o momento corrente	SeriesInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SERIES_FIRSTDATE	A primeira data para o ativo-período para o momento corrente	SeriesInfoInteger
SERIES_LASTBAR_DATE	Hora de abertura da última barra do ativo-período	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	A primeira data no histórico do ativo no servidor independentemente da janela de tempo	SeriesInfoInteger
SERIES_SYNCHRONIZED	Flag de sincronização de dados de um ativo/período para o momento corrente	SeriesInfoInteger
SERIES_TERMINAL_FIRSTDATE	A primeira data no histórico	SeriesInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	do ativo no terminal cliente, independentemente da janela de tempo	
SHORT_MAX	Valor máximo, que pode ser representado pelo tipo short	Constantes de Tipo Numérico
SHORT_MIN	Valor mínimo, que pode ser representado pelo tipo short	Constantes de Tipo Numérico
SIGNAL_BASE_AUTHOR_LOGIN	Author login	SignalBaseGetString
SIGNAL_BASE_BALANCE	Account balance	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Broker name (company)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	Broker server	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Signal base currency	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Publication date (date when it	SignalBaseGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	become available for subscription)	
SIGNAL_BASE_DATE_STARTED	Monitoring starting date	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Account equity	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Account gain	SignalBaseGetDouble
SIGNAL_BASE_ID	Signal ID	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Account leverage	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Account maximum drawdown	SignalBaseGetDouble
SIGNAL_BASE_NAME	Signal name	SignalBaseGetString
SIGNAL_BASE_PIPS	Profit in pips	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Signal subscription price	SignalBaseGetDouble
SIGNAL_BASE_RATING	Position in rating	SignalBaseGetInteger
SIGNAL_BASE_ROI	Return on Investment (%)	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Number of subscribers	SignalBaseGetInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SIGNAL_BASE_TRADE_MODE	Account type (0-real, 1-demo, 2-contest)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Number of trades	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	The flag enables synchronization without confirmation dialog	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	Copy Stop Loss and Take Profit flag	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Deposit percent (%)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Equity limit	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	Signal id, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Signal name, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Slippage (used when placing market orders in synchronization of position	SignalInfoGetDouble, SignalInfoSetDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	s and copying of trades)	
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"Copy trades by subscription" permission flag	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	"Agree to terms of use of Signals service" flag, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Maximum percent of deposit used (%), r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	Linha de sinal	Linhas de Indicadores
STAT_BALANCE_DD	Máxima queda de saldo em termos monetários Em um processo de negociação, um saldo pode sofrer numerosas quedas; aqui o	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	maior valor é tomado	
STAT_BALANCE_DD_RELATIVE	Queda de saldo em termos monetários que foi registrada no momento da máxima queda do saldo como uma percentagem (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics
STAT_BALANCE_DDREL_PERCENT	Máxima queda de saldo com uma percentagem. Em um processo de negociação, um saldo pode sofrer numerosas quedas, para cada	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	qual a valor da queda relativa em percentagem é calculada. O maior valor é retornado	
STAT_BALANCEDD_PERCENT	Queda de saldo como uma percentagem que foi registrada no momento da máxima queda do saldo em termos monetários (STAT_BALANCE_DD).	TesterStatistics
STAT_BALANCEMIN	Valor do saldo mínimo	TesterStatistics
STAT_CONLOSSMAX	Prejuízo máximo em uma série de negociações perdedoras. O valor é	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	menor ou igual a zero	
STAT_CONLOSSMAX_TRADES	O número de negociações (trades) que formaram oSTAT_CONLOSSMAX(máximo prejuízo em uma série de negociações perdedoras)	TesterStatistics
STAT_CONPROFITMAX	Lucro máximo em uma série de negociações lucrativas. O valor é maior ou igual a zero	TesterStatistics
STAT_CONPROFITMAX_TRADES	O número de negociações que formaram oSTAT_CONPROFITMAX(lucro	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	máximo em uma série de negociações lucrativas)	
STAT_CUSTOM_ONTESTER	O valor do critério de otimização customizado calculado retornado através da função OnTester()	TesterStatistics
STAT_DEALS	Número de operações (deals)	TesterStatistics
STAT_EQUITY_DD	Máxima queda do saldo a mercado em termos monetários. Em um processo de negociação, numerosas quedas	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	podem ocorrer no saldo a mercado; aqui o maior valor é tomado	
STAT_EQUITY_DD_RELATIVE	Queda do saldo a mercado em termos monetários que foi registrada no momento da máxima queda do saldo a mercado em percentagem (STAT_EQUITY_DDREL_PERCENT).	TesterStatistics
STAT_EQUITY_DDREL_PERCENT	Máxima queda do saldo a mercado como uma percentagem. Em um processo de	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	negociação, um saldo a mercado pode sofrer numerosas quedas, para cada qual o valor da queda relativa em percentagem é calculado. O maior valor é retornado	
STAT_EQUITYDD_PERCENT	Queda em percentagem que foi registrada no momento da máxima queda do saldo a mercado em termos monetários (STAT_EQUITY_DD).	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
STAT_EQUITYMIN	Mínimo valor do saldo a mercado	TesterStatistics
STAT_EXPECTED_PAYOFF	Recompensa esperada	TesterStatistics
STAT_GROSS_LOSS	Prejuízo total, a soma de todas as negociações negativas. O valor é menor ou igual a zero	TesterStatistics
STAT_GROSS_PROFIT	Lucro total, a soma de todas as negociações lucrativas (positivas). O valor é maior ou igual a zero	TesterStatistics
STAT_INITIAL_DEPOSIT	O valor do depósito inicial	TesterStatistics
STAT_LONG_TRADES	Negociações Compradas	TesterStatistics
STAT_LOSS_TRADES	Negociações	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	perdedoras	
STAT_LOSSTRADES_AVGCON	Comprimento médio de uma série de negociações perdedores	TesterStatistics
STAT_MAX_CONLOSS_TRADES	O número de negociações na mais longa série de negociações perdedoras STAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	O total de prejuízo da mais longa série de negociações perdedoras	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	O número de negociações na mais longa série de negociaç	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ões lucrativasSTAT_MAX_CONWINS	
STAT_MAX_CONWINS	O lucro total da mais longa série de negociações lucrativas	TesterStatistics
STAT_MAX_LOSSTRADE	Prejuízo Máximo - o menor valor dentre todos as negociações (trades) perdedoras O valor é menor ou igual a zero	TesterStatistics
STAT_MAX_PROFITTRADE	Lucro máximo - o maior valor dentre todas as negociações lucrativas O valor é maior ou igual a zero	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
STAT_MIN_MARGINLEVEL	Valor mínimo do nível de margem	TesterStatistics
STAT_PROFIT	Lucro líquido após o teste, a soma de STAT_GROSS_PROFIT e STAT_GROSS_LOSS (STAT_GROSS_LOSS é sempre menor ou igual a zero)	TesterStatistics
STAT_PROFIT_FACTOR	Fator de lucro, igual a razão de STAT_GROSS_PROFIT/STAT_GROSS_LOSS. Se STAT_GROSS_LOSS=0, o fator de lucro é igual aDBL_MAX	TesterStatistics
STAT_PROFIT_LONGTRADES	Negociações compradas	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	lucrativas	
STAT_PROFIT_SHORTTRADES	Negociações vendidas lucrativas	TesterStatistics
STAT_PROFIT_TRADES	Negociações compradas lucrativas	TesterStatistics
STAT_PROFITTRADES_AVGCON	Comprimento médio de uma série de negociações lucrativas	TesterStatistics
STAT_RECOVERY_FACTOR	Fator de recuperação, igual a razão de STAT_PROFIT/STAT_BALANCE_DD	TesterStatistics
STAT_SHARPE_RATIO	Razão Sharpe	TesterStatistics
STAT_SHORT_TRADES	Negociações Vendidas	TesterStatistics
STAT_TRADES	Número de negociações (trades)	TesterStatistics

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
STAT_WITHDRAWAL	Dinheiro sacado da conta	TesterStatistics
STO_CLOSECLOSE	Cálculo é baseado em preços Fechamento/Fechamento	Constantes de Preço
STO_LOWHIGH	Cálculo é baseado em preços Mínimo/Máximo	Constantes de Preço
STYLE_DASH	Linha tracejada	Estilos de Desenho
STYLE_DASHDOT	Linha traço-ponto	Estilos de Desenho
STYLE_DASHDOTDOT	Traço - dois pontos	Estilos de Desenho
STYLE_DOT	Linha pontilhada	Estilos de Desenho
STYLE_SOLID	Linha sólida	Estilos de Desenho
SUNDAY	Domingo	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Preço de Compra (Ask) - melhor oferta de compra	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_ASKHIGH	Máximo Preço de Compra do dia	SymbolInfoDouble
SYMBOL_ASKLOW	Mínimo Preço de Compra do dia	SymbolInfoDouble
SYMBOL_BANK	Fornecedor da cotação corrente	SymbolInfoString
SYMBOL_BASIS	The underlying asset of a derivative	SymbolInfoString
SYMBOL_BID	Preço de Venda (Bid) - melhor oferta de venda	SymbolInfoDouble
SYMBOL_BIDHIGH	Máximo Preço de Venda do dia	SymbolInfoDouble
SYMBOL_BIDLOW	Mínimo Preço de Venda do dia	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	Modo CFD - cálculo de margem e lucro para CFD	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	Modo índice	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	CFD - cálculo de margem e lucro para CFD por índices	
SYMBOL_CALC_MODE_CFDLEVERAGE	Modo Alavancado CFD - cálculo de margem e lucro para CFD em negociação alavancada	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Modo Futures - cálculo da margem e lucro para os contratos futuros de negociação na bolsa de valores	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	Modo FORTS Futures - cálculo de margem e lucro para futuros contratos de	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	comercialização nos FORTS.	
SYMBOL_CALC_MODE_EXCH_STOCKS	Modo Exchange - cálculo de margem e lucro para negociação de títulos na bolsa de valores	SymbolInfoInteger
SYMBOL_CALC_MODE_FOREX	Modo Forex - cálculo de lucro e margem para Forex	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Modo Futuros - cálculo de margem e lucro para futuros	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-tradable asset on a trading account.	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_CURRENCY_BASE	Moeda base de um ativo	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	Moeda de margem	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	Moeda de lucro	SymbolInfoString
SYMBOL_DESCRIPTION	Descrição do ativo	SymbolInfoString
SYMBOL_DIGITS	Dígitos após um ponto decimal	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	A ordem é válida até o fim do dia	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	A ordem é válida durante um período de tempo ilimitado, até que ela seja explicitamente cancelada	SymbolInfoInteger
SYMBOL_EXPIRATION_MODE	Flags de modos de expiração de ordens permitidas	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_EXPIRATION_SPECIFIED	A hora de expiração é especificada na ordem	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	A data de expiração é especificada na ordem	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	Data do fim de negociação do ativo (geralmente usado para futuros)	SymbolInfoInteger
SYMBOL_FILLING_FOK	Esta política significa que uma transação somente pode ser executada com o volume especificado. Se a quantidade desejada do ativo não está disponív	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	el no mercado, a ordem não será executada. O volume requerido pode ser preenchido usando várias ofertas disponíveis no mercado no momento.	
SYMBOL_FILLING_IOC	Neste caso, o investidor concorda em executar uma transação com o volume máximo disponível no mercado, em que foi indicado na ordem. No caso da ordem não poder	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ser completamente preenchida, o volume disponível da ordem será executado, e o volume remanescente será cancelado. A possibilidade de utilização de ordens "IOC" é determinado no servidor.	
SYMBOL_FILLING_MODE	Flags de modos de preenchimento de ordens permitidas	SymbolInfoInteger
SYMBOL_ISIN	O nome de um ativo no sistema ISIN (International Securities	SymbolInfoString

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	Identification Number). O International Securities Identification Number é um código de 12 dígitos alfanumérico que identifica de forma única um ativo financeiro. A presença desta propriedade de ativo é determinada no lado do servidor de negociação.	
SYMBOL_LAST	Preço da último operação (deal)	SymbolInfoDouble
SYMBOL_LASTHIGH	Máximo dos preços negociad	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	os do dia	
SYMBOL_LASTLOW	Mínimo dos preços negociados do dia	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	Margem inicial significa a quantia de moeda de margem requerida para abrir uma posição com o volume de um lote. É usada para verificar os ativos de um cliente quando ele ou ele entra no mercado.	SymbolInfoDouble
SYMBOL_MARGIN_MAINTENANCE	A margem de manutenção. Se for definida	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	, ele representa a quantia de margem na moeda de margem do ativo cobrada de um lote. É usada parar verificar os ativos de um cliente quando o estado da conta dele/dela muda. Se a margem de manutenção é igual a 0, a margem inicial é usada.	
SYMBOL_OPTION_MODE	Option type	SymbolInfoInteger
SYMBOL_OPTION_MODE_AMERICAN	American option may be exercised on any trading day on or	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	before expiry. The period within which a buyer can exercise the option is specified for it	
SYMBOL_OPTION_MODE_EUROPEAN	European option may only be exercised on a specified date (expiration, execution date, delivery date)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	Option right (Call/Put)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	A call option gives you the right to buy an asset at a specified price	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_PUT	A put option gives you the	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	right to sell an asset at a specified price	
SYMBOL_OPTION_STRIKE	The strike price of an option. The price at which an option buyer can buy (in a Call option) or sell (in a Put option) the underlying asset, and the option seller is obliged to sell or buy the appropriate amount of the underlying asset.	SymbolInfoDouble
SYMBOL_ORDER_LIMIT	Ordens de limite são permitidas (Buy Limit e	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	Sell Limit)	
SYMBOL_ORDER_MARKET	Ordens de mercado são permitidas (Buy e Sell)	SymbolInfoInteger
SYMBOL_ORDER_MODE	Flags de modos de preenchimento de ordens permitidas	SymbolInfoInteger
SYMBOL_ORDER_SL	Stop Loss é permitido	SymbolInfoInteger
SYMBOL_ORDER_STOP	Ordens de parada são permitidas (Buy Stop e Sell Stop)	SymbolInfoInteger
SYMBOL_ORDER_STOP_LIMIT	Ordens de limite de parada são permitidas (Buy Stop Limit e Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	Take Profit é	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	permitido	
SYMBOL_PATH	Caminho da árvore de ativos	SymbolInfoString
SYMBOL_POINT	Valor de um ponto do ativo	SymbolInfoDouble
SYMBOL_SELECT	O ativo (símbolo) é selecionado na janela Observação de Mercador	SymbolInfoInteger
SYMBOL_SESSION_AW	Preço médio ponderado da sessão atual	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	Número de ordens de Compra no momento	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Volume atual de ordens de Compra	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Preço de fechamento da	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	sessão atual	
SYMBOL_SESSION_DEALS	Número de operações (deals) na sessão corrente	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Juros de abertura resumo	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Preço de abertura da sessão atual	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	Máximo preço da sessão atual	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	Mínimo preço da sessão atual	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Preço de liquidação da sessão atual	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Número de ordens de Venda no momento	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Volume atual de ordens	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	de Venda	
SYMBOL_SESSION_TURNOVER	Rotatividade resumo da sessão corrente	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Volume resumo de operações da sessão corrente	SymbolInfoDouble
SYMBOL_SPREAD	Valor do spread em pontos	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	Indicação de um spread flutuante	SymbolInfoInteger
SYMBOL_START_TIME	Data do início de negociação do ativo (geralmente usado para futuros)	SymbolInfoInteger
SYMBOL_SWAP_LONG	Valor de swap comprado	SymbolInfoDouble
SYMBOL_SWAP_MODE	Modelo de cálculo de Swap	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swaps são cobrados em dinheiro na moeda de depósito do cliente	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swaps são cobrados em dinheiro na moeda de margem do ativo	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swaps são cobrados em dinheiro na moeda base do ativo	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	Desabilita swaps (sem swaps)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swaps são cobrados como juro anual especificado a partir do preço de instrume	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	nto no cálculo do swap (ano bancário padrão é 360 dias)	
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swaps são cobradas como juro anual especificado a partir do preço de abertura da posição (ano bancário padrão é 360 dias)	SymbolInfoInteger
SYMBOL_SWAP_MODE_POINTS	Swaps são cobrados em pontos	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	Swaps são cobradas através de posições reabertas Ao final de um dia de negociação a posição é	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	encerrada. No dia seguinte ela é reaberta através do preço de Venda (Bid) corrente +/- um número específico de pontos (parâmetros SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swaps são cobradas através de posições reabertas Ao final de um dia de negociação a posição é encerrada. No dia seguinte ela é reaberta através	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	do preço de fechamento +/- um número específico de pontos (parâmetros SYMBOL_SWAP_LONG e SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_ROLLOVER3DAYS	Dia da semana para cobrar rolagem de swap de 3 dias	SymbolInfoInteger
SYMBOL_SWAP_SHORT	Valor de swap vendido	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Número máximo de solicitações exibidas na Profundidade do Mercado. Para ativos que não tem fila de solicitações, o valor é	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	igual a zero.	
SYMBOL_TIME	Hora da última cotação	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	Modo de cálculo de preço de contrato	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Tamanho do contrato de negociação	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	Execução em um sistema de negociação externo (Exchange)	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Execução instantânea	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	Execução a Mercado	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	Execução por solicitação	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	Modo de execução de operação (deal)	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_TRADE_FREEZE_LEVEL	Distância para congelar operações de negociação em pontos	SymbolInfoInteger
SYMBOL_TRADE_MODE	Tipo de execução de ordem	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	Permitido somente operações de fechamento de posição	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	Negociação está desabilitada para o ativo	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Sem restrições de negociação	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	Permitido somente posições compradas	SymbolInfoInteger
SYMBOL_TRADE_MODE_SHORTONLY	Permitido somente posições vendidas	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	Mínima distânci	SymbolInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	a em pontos a partir do preço de fechamento corrente para colocar ordens de Stop	
SYMBOL_TRADE_TICK_SIZE	Mínima mudança de preço	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	Valor do SYMBOL_TRADE_TICK_VALUE_PROFIT	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	Preço de tick calculado para um posição perdedora	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	Preço de tick calculado para uma posição lucrativa	SymbolInfoDouble
SYMBOL_VOLUME	Volume da última operação (deal)	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Máximo volume agregado	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	permitido de uma posição aberta e ordens pendentes em uma direção (compra ou venda) para um ativo. Por exemplo, com a limitação de 5 lotes, você pode ter uma posição comprada aberta com o volume de 5 lotes e colocar uma ordem pendente de Sell Limit com o volume de 5 lotes. Mas neste caso você não pode colocar um	

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ordem pendente de Buy Limite (já que o total de volume em uma direção excederia o limite) ou colocar Sell Limit com o volume maior que 5 lotes.	
SYMBOL_VOLUME_MAX	Máximo volume para uma operação (deal)	SymbolInfoDouble
SYMBOL_VOLUME_MIN	Mínimo volume para uma operação (deal)	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Mínimo passo de mudança de volume para execução de uma operação (deal)	SymbolInfoDouble

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
SYMBOL_VOLUMEHIGH	Volume diário máximo	SymbolInfoInteger
SYMBOL_VOLUMELOW	Volume diário mínimo	SymbolInfoInteger
TENKANSEN_LINE	Linha Tenkan-sen	Linhas de Indicadores
TERMINAL_BUILD	O número de compilação (build) do terminal cliente	TerminalInfoInteger
TERMINAL_CODEPAGE	Número do código de página da linguagem instalado no terminal cliente	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	Caminho comum para todos os terminais instalados em um computador	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	The flag indicates the	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	presence of MQL5.community authorization data in the terminal	
TERMINAL_COMMUNITY_BALANCE	Balance in MQL5.community	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	Connection to MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Nome da empresa	TerminalInfoString
TERMINAL_CONNECTED	Conexão com um servidor de negociação	TerminalInfoInteger
TERMINAL_CPU_CORES	O número de núcleos de CPU no sistema	TerminalInfoInteger
TERMINAL_DATA_PATH	Pasta na qual dados do terminal são armazenados	TerminalInfoString
TERMINAL_DISK_SPACE	Espaço livre de disco para a	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	pasta MQL5\Files do terminal (agente), MB	
TERMINAL_DLLS_ALLOWED	Permissão para usar DLL	TerminalInfoInteger
TERMINAL_EMAIL_ENABLED	Permissão para enviar e-mails usando servidor-SMTP e login, especificado nas configurações do terminal	TerminalInfoInteger
TERMINAL_FTP_ENABLED	Permissão para enviar relatórios usando servidor-FTP e login, especificado nas configurações do terminal	TerminalInfoInteger
TERMINAL_LANGUAGE	Linguagem do terminal	TerminalInfoString
TERMINAL_MAXBARS	A quantidade máxima de barras	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	no gráfico	
TERMINAL_MEMORY_AVAILABLE	Memória livre do processo do terminal (agente), MB	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	Memória física no sistema, MB	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	Memória disponível para o processo do terminal (agente), MB	TerminalInfoInteger
TERMINAL_MEMORY_USED	Memória usada pelo terminal (agente), MB	TerminalInfoInteger
TERMINAL_MQID	The flag indicates the presence of MetaQuotes ID data to send Push notifications	TerminalInfoInteger
TERMINAL_NAME	Nome do terminal	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Permission to	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	send notifications to smartphone	
TERMINAL_OPENCL_SUPPORT	A versão do OpenCL suportado no formato 0x00010002 = 1.2. "0" significa que OpenCL não é suportado	TerminalInfoInteger
TERMINAL_PATH	Pasta a partir da qual o terminal foi iniciado	TerminalInfoString
TERMINAL_PING_LAST	Último valor conhecido do ping até ao servidor de negociação em microssegundos. Em um segundo existe um milhão de microssegundos	TerminalInfoInteger

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TERMINAL_SCREEN_DPI	A qualidade de resolução ao exibir informações no ecrã é medida pelo número de pontos por polegada linear da superfície (DPI)	TerminalInfoInteger
TERMINAL_TRADE_ALLOWED	Permissão para negociar	TerminalInfoInteger
TERMINAL_X64	Indicação do "terminal 64-bit"	TerminalInfoInteger
THURSDAY	Quinta-feira	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Coloca um ordem de negociação para execução imediata com os parâmetros específicos (ordem a	MqlTradeRequest

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	mercado)	
TRADE_ACTION_MODIFY	Modifca os parâmetros de uma ordem coloca anteriormente	MqlTradeRequest
TRADE_ACTION_PENDING	Coloca um ordem de negociação para execução sob condições específicas (ordem pendente)	MqlTradeRequest
TRADE_ACTION_REMOVE	Exclui uma ordem pendente colocada anteriormente	MqlTradeRequest
TRADE_ACTION_SLTP	Modifica valores de Stop Loss e Take Profit de uma posição em aberto	MqlTradeRequest

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TRADE_RETCODE_CANCEL	Solicitação cancelada pelo negociador	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	Autotrading desabilitado pelo terminal cliente	MqlTradeResult
TRADE_RETCODE_CONNECTION	Sem conexão com o servidor de negociação	MqlTradeResult
TRADE_RETCODE_DONE	Solicitação concluída	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	Somente parte da solicitação foi concluída	MqlTradeResult
TRADE_RETCODE_ERROR	Erro de processamento de solicitação	MqlTradeResult
TRADE_RETCODE_FROZEN	Ordem ou posição congeladas	MqlTradeResult
TRADE_RETCODE_INVALID	Solicitação inválida	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TRADE_RETCODE_INVALID_EXPIRATION	Data de expiração da ordem inválida na solicitação	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	Tipo de preenchimento de ordem inválido	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	Incorreto ou proibida tipo de ordem	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Preço inválido na solicitação	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	Stops inválidos na solicitação	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	Volume inválido na solicitação	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	O número de ordens pendentes atingiu o limite	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TRADE_RETCODE_LIMIT_VOLUME	O volume de ordens e posições para o ativo atingiu o limite	MqlTradeResult
TRADE_RETCODE_LOCKED	Solicitação bloqueada para processamento	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Mercado está fechado	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	Sem alterações na solicitação	MqlTradeResult
TRADE_RETCODE_NO_MONEY	Não existe dinheiro suficiente para completar a solicitação	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	Operação é permitida somente para contas reais	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	Estado da ordem alterada	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TRADE_RETCODE_PLACED	Ordem colocada	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	Posição com a especificação POSITION_IDENTIFIER já foi fechado	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Os preços se alteraram	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	Não existem cotações para processar a solicitação	MqlTradeResult
TRADE_RETCODE_REJECT	Solicitação rejeitada	MqlTradeResult
TRADE_RETCODE_REQUOTE	Nova Cotação	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	Autotrading desabilitado pelo servidor	MqlTradeResult
TRADE_RETCODE_TIMEOUT	Solicitação cancelada por expiração de tempo	MqlTradeResult

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	(timeout)	
TRADE_RETCODE_TOO_MANY_REQUESTS	Solicitações com freqüência em excesso	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Negociação está desabilitada	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	Adição de uma operação (deal) para o histórico. A ação é realizada como resultado de uma execução de uma ordem ou realização de operações com o saldo da conta.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	Exclusão de uma operação (deal) do histórico. Pode haver casos quando	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	uma operação (deal) executada anteriormente é excluída de um servidor. Por exemplo, uma operação (deal) foi excluída de um sistema de negociação externo (exchange) onde ela foi transferida anteriormente por uma corretora (broker).	
TRADE_TRANSACTION_DEAL_UPDATE	Atualização de uma operação (deal) no histórico. Pode haver caso quando uma	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	operação (deal) executada previamente é alterada em um servidor. Por exemplo, uma operação (deal) foi alterada em um sistema de negociação externo (exchange) onde ela foi transferida anteriormente por uma corretora (broker).	
TRADE_TRANSACTION_HISTORY_ADD	Adição de uma ordem no histórico como resultado de uma execução ou cancelamento.	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
TRADE_TRANSACTION_HISTORY_DELETE	Exclusão de uma ordem do histórico de ordens. Este tipo é fornecido para aumentar a funcionalidade no lado de um servidor de negociação.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_UPDATE	Alteração de uma ordem localizada no histórico de ordens. Este tipo é fornecido para aumentar a funcionalidade no lado de um servidor de negociação.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	Adição de uma	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	nova ordem de abertura.	
TRADE_TRANSACTION_ORDER_DELETE	Remoção de uma ordem da lista de ordens em aberto. Uma ordem pode ser excluída da lista de ordens em aberto como resultado da configuração de um solicitação apropriada ou execução (preenchimento) ou movimentação para o histórico.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_UPDATE	Atualização de uma	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	ordem de abertura r. As atualizações incluem não somente mudanças evidentes provenientes do terminal cliente ou do lado de um servidor de negociação, mas também alterações de estado de uma ordem ao configurá-lo (por exemplo, transição de ORDER_STATE_STARTED para ORDER_STATE_PLACED ou de ORDER_STATE_PLACED	

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	para ORDER_STATE_PARTIAL, etc.).	
TRADE_TRANSACTION_POSITION	Alteração de uma posição não relacionada com a execução de uma operação (deal). Este tipo de transação mostra que uma posição foi alterada pelo lado de um servidor de negociação. O volume de uma posição, o preço de abertura, os níveis de Stop Loss e Take Profit podem ter sido alterado	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	s. Dados sobre alteração são submetidos na estrutura MqlTradeTransaction via o handler OnTradeTransaction. Alterações de posição (adição, alteração ou encerramento), como resultado de uma execução de operação (deal), não levam a ocorrência de uma transação TRADE_TRANSACTION_POSITION.	
TRADE_TRANSACTION_REQUEST	Notificação do fato de	MqlTradeTransaction

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	que uma solicitação de negociação foi processada por um servidor e o resultado processado foi recebido. Apenas o campo type (tipo de transação de negociação) deve ser analisado em tais transações na estrutura MqlTradeTransaction. O segundo e terceiro parâmetros de OnTradeTransaction (request e result) devem ser analisados para	

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	dados adicionais.	
TUESDAY	Terça-feira	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	Valor máximo, que pode ser representado pelo tipo uchar	Constantes de Tipo Numérico
UINT_MAX	Valor máximo, que pode ser representado pelo tipo uint	Constantes de Tipo Numérico

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
ULONG_MAX	Valor máximo, que pode ser representado pelo tipo ulong	Constantes de Tipo Numérico
UPPER_BAND	Limite superior	Linhas de Indicadores
UPPER_HISTOGRAM	Histograma superior	Linhas de Indicadores
UPPER_LINE	Linha de cima	Linhas de Indicadores
USHORT_MAX	Valor máximo, que pode ser representado pelo tipo ushort	Constantes de Tipo Numérico
VOLUME_REAL	Volume de negociação	Constantes de Preço
VOLUME_TICK	Volume de Tick	Constantes de Preço
WEDNESDAY	Quarta-feira	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Significa o número de itens restantes até o final do array, isto é, o array inteiro	Outras Constantes

© 2000-2024, MetaQuotes Ltd.

Constant	Description	Usage
	será processado	
WRONG_VALUE	A constante pode ser implicitamente convertido para qualquer tipo de enumeração	Outras Constantes

© 2000-2024, MetaQuotes Ltd.